One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

ADP8860 Software User Guide

INTRODUCTION

This user guide describes the functionality of the Analog Devices, Inc., ADP8860 and provides software development guidelines. The ADP8860 communicates with an external processor using an I^2C interface and an interrupt line (nINT). The processor sends initialization and activation commands to the ADP8860, which acts as a slave device.

The interrupt line, from the ADP8860 to the processor, is used to indicate a failure condition, such as a thermal shutdown or an overvoltage and LED/output short circuit, or to indicate a light level threshold has been crossed. All interrupt sources are maskable. Refer to Figure 1 for a typical application diagram. Figure 2 shows a schematic with keypad light control.

The interrupt line is active low; each interrupt source has an individual masking bit. The processor can reset the ADP8860 anytime by pulling the nRST line low; this operation reinitializes the ADP8860 at the default state and places the device in standby mode.

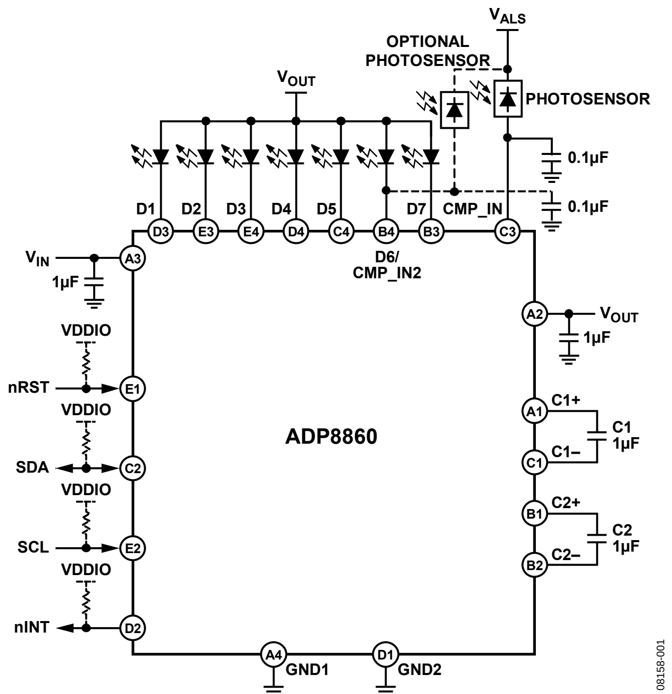

Figure 1. Typical Application Schematic with Optional Second Photo Diode

TABLE OF CONTENTS

REVISION HISTORY

7/09—Revision 0: Initial Version

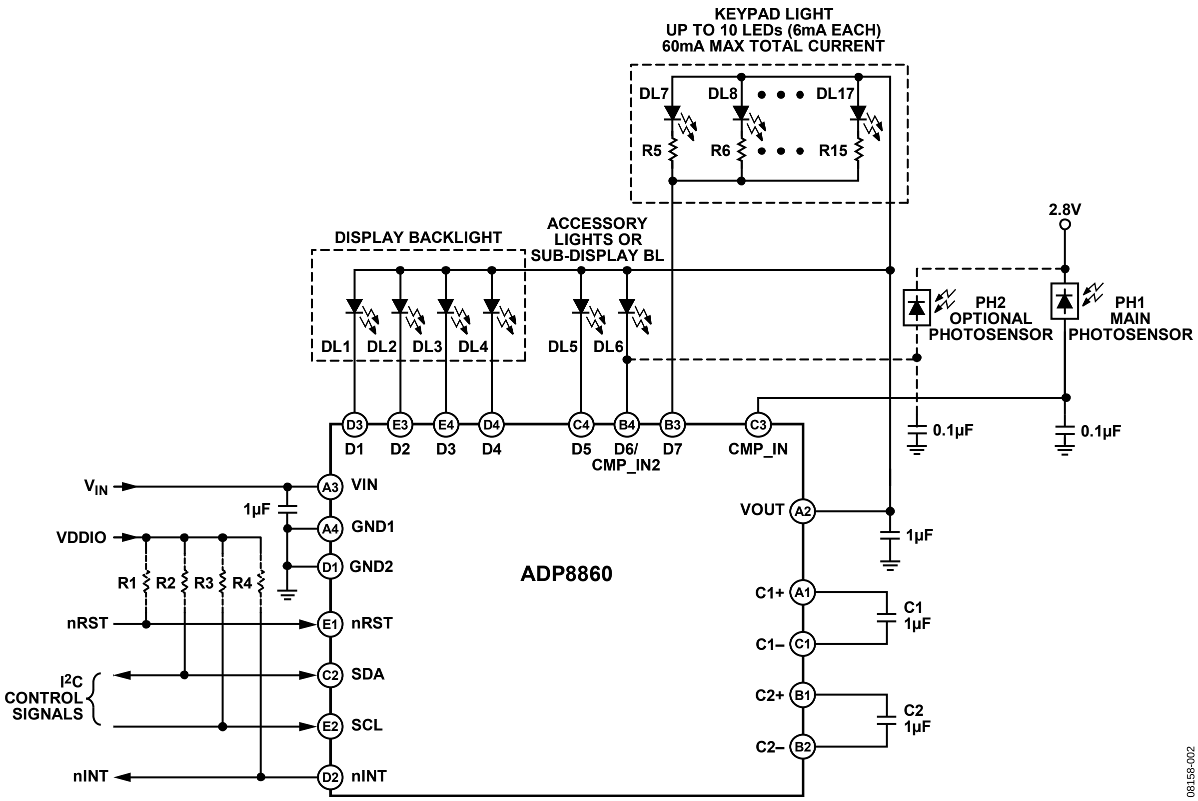

Figure 2. Schematic with Keypad Light Control

I^{2}C INTERFACE MODE

The ADP8860 includes an I^{2}C-compatible serial interface for controlling the LED current, as well as for readback of system status registers. The I^{2}C device address is 0x54 (0101 0100, binary) for a write sequence and 0x55 (0101 0101, binary) for a read sequence. Figure 3 shows the I^{2}C write sequence while Figure 4 shows a read operation.

The ADP8860 sends data from the register denoted by the register address. The lowest bit number (0) represents the least significant bit, and the highest bit number (7) represents the most significant bit. The register address content selects which of the ADP8860 registers data is written to or read from.

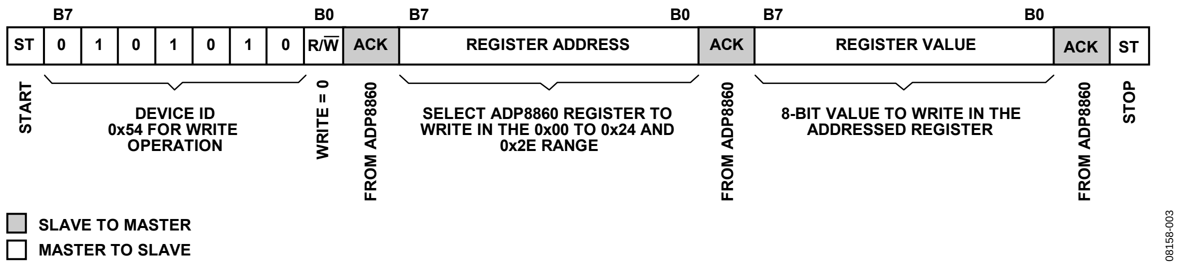

Figure 3. I^{2}C Write Operation

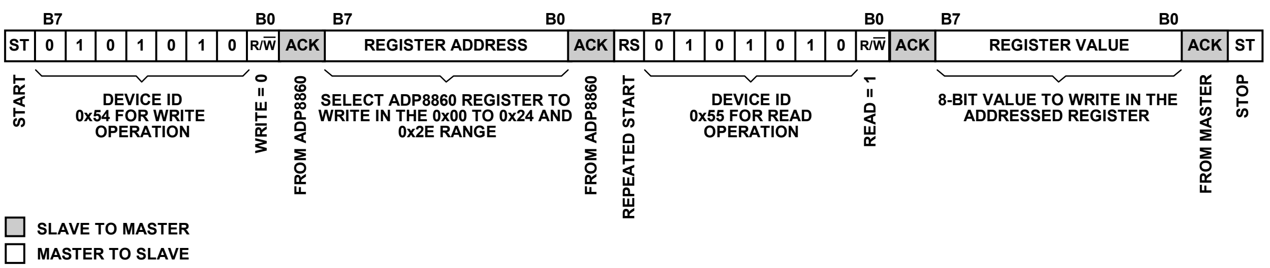

Figure 4. I^{2}C Read Operation

INTERRUPTS

There are up to five interrupt sources available on the ADP8860 as follows:

- CMP_INT is from the main light sensor comparator.
- CMP2_INT is from sensor Comparator 2.
- OVP_INT is from the overvoltage protection comparator.
- TSD_INT is from the thermal shutdown circuit.
- SHORT_INT is from the short-circuit detection comparator.

Each interrupt has individual masking/enable bits mapped in Register INTR_EN. If the respective bit in the masking register is 0, an interrupt is not generated to the external processor, however the interrupt pending bit (on the MDCR2 register) can be set in case the monitored condition occurs. This can be used by the processor to periodically poll the interrupt pending register (Register MDCR2) looking for an event to be true. If the masking register bit is 1 an interrupt is generated (the nINT line goes low) to the external processor in case an interrupt pending bit is set.

CMP_INT is set every time the main light sensor comparator detects a threshold transition (rising or falling condition). This comparator has two programmable thresholds (L2 and L3) defining the transition level from dark to office (L3) and from office to outdoor (L2).

The CMP2_INT interrupt works the same way as CMP_INT except that the sensing input comes from the second light sensor. The programmable threshold is the same as the main light sensor comparator.

The OVP_INT interrupt is generated when the charge pump output voltage rises above a safety limit. In the event of an over-voltage condition, the charge-pump is disabled until the output voltage decreases to a recovery working level. An overvoltage event can be generated when the load is removed from the circuit and the input voltage, multiplied by the charge pump gain (can be 1.5× or 2×), is above the OVP limit. In this condition, the interrupt to the external processor is generated periodically. Therefore, the software should handle a case of this sort by turning off the ADP8860 or disabling the OVP interrupt mask.

The TSD_INT interrupt is generated when the die temperature in the ADP8860 rises above a safety limit, typically 150°. When this condition occurs, the charge pump and the LED drivers are turned off waiting for the die to cool down. When the die temperature decreases below ~130° the circuit is activated again automatically. No interrupt is generated when the device turns on again, however if the software clears the pending interrupt and the temperature remains above 130° another interrupt is generated.

The SHORT_INT interrupt is generated if the output of the charge-pump VOUT is shorted to ground or an LED connected to the sink output is shorted. In a short-circuit event, the charge pump and drivers are turned off immediately and the short-circuit pending flag is set. The processor may then retry new activations or issue a diagnostic message. Interrupts are cleared by writing a 1 in the pending register bit. If Bit INT_CFG in Register MDCR is set to 1, this forces the nINT line deassertion (Logic High) for 50 µs after the processor clears the interrupt pending, and the interrupt condition persists. If INT_CFG is 0, the nINT line remains asserted (Logic Low) after the processor clears the interrupt pending and the interrupt condition persists.

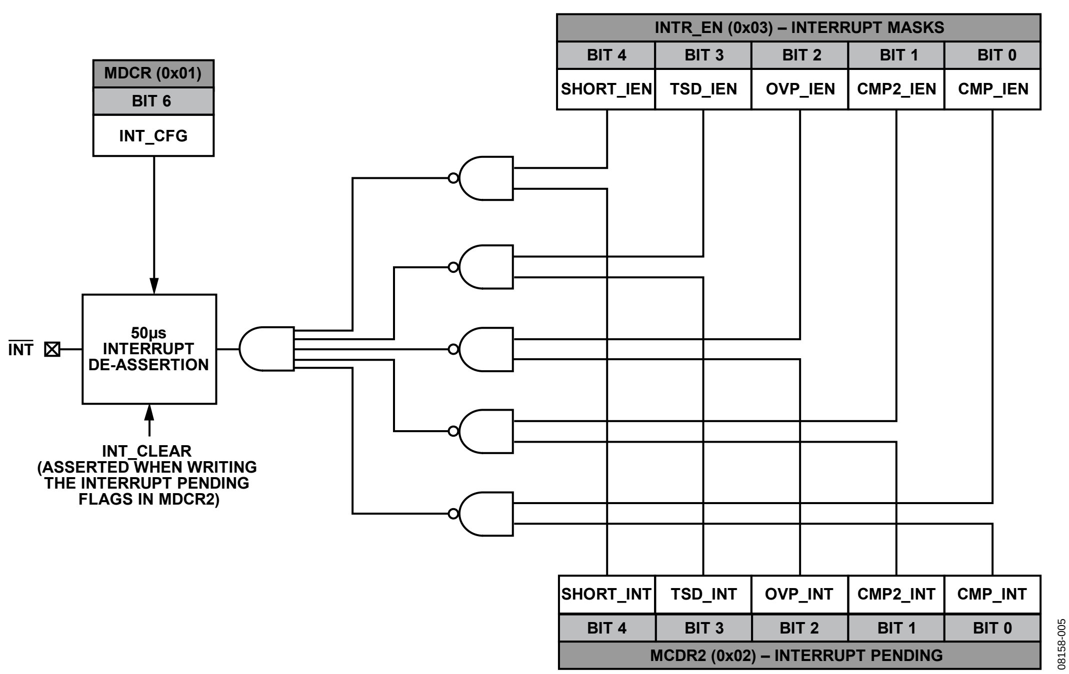

Figure 5. Interrupt Logic

DATA FLOW DIAGRAMS

Figure 6 and Figure 7 show the backlight data flow and the individual data flow, respectively.

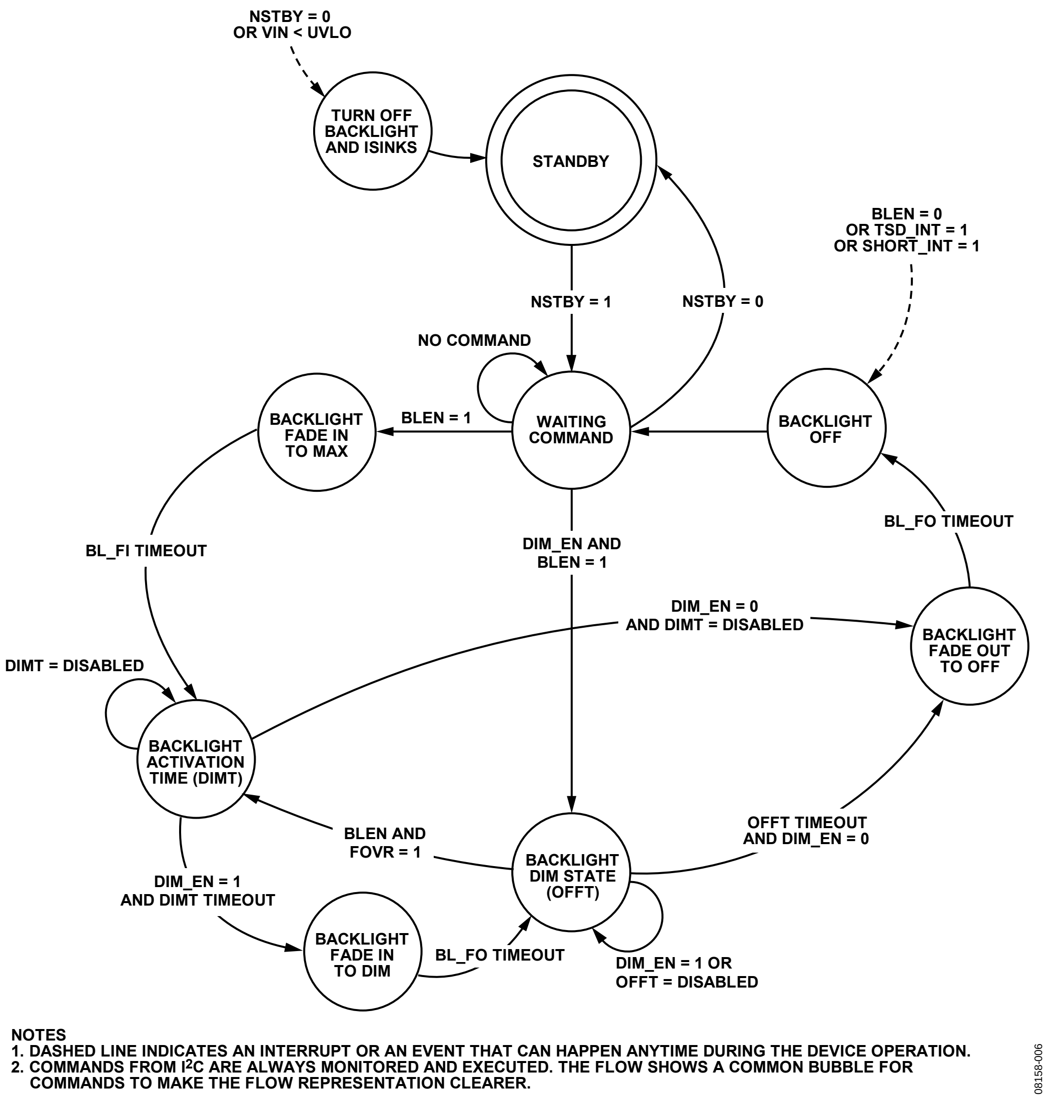

Figure 6. Backlight Data Flow

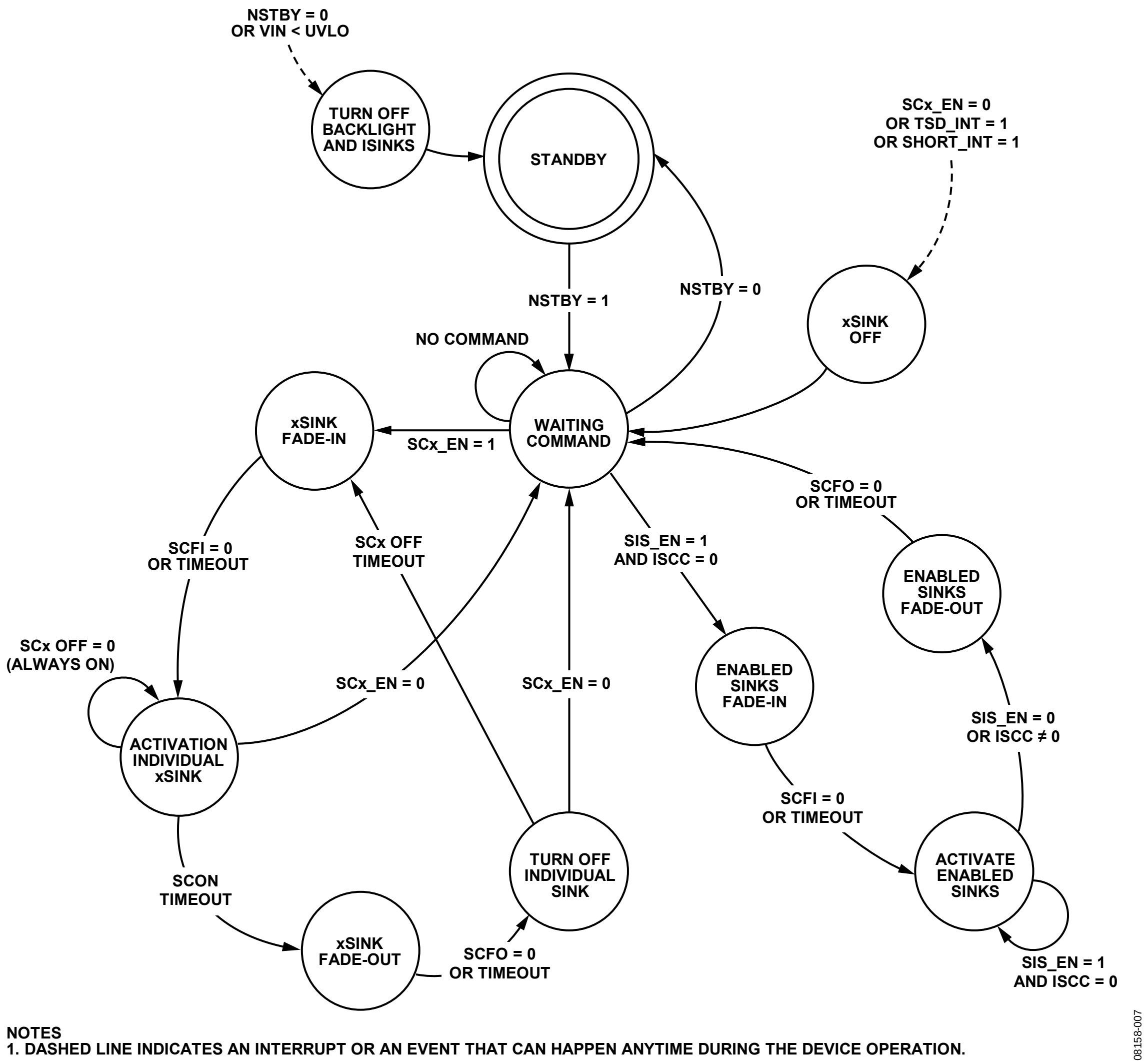

Figure 7. Individual Sinks Data Flow

SETTING THE LED CURRENT

The ADP8860 contains seven current sink outputs that can be software configured to operate as part of the LED backlight or as individual sinks, such as funlights or keypad lights. All outputs can drive up to 30 mA with the exception of the seventh sink (D1B/LED7) that can drive up to 60 mA, which is ideal for keypad light applications where up to 10 LEDs can be driven in parallel with shunt resistors (see Figure 2).

The application software must program which output to use as part of the backlight and which to use for the individual sinks. Register BLSEN is used for this purpose. If the selection bit associated with the LED sink (Bit 0 controls LED1, Bit 1 controls LED2, and so on) is set to 0, it is part of the LED backlight; otherwise, if set to 1, it is an independent sink. In this case, Register ISCC controls the individual sink turn-on and turn-off activations. For example, if BLSEN is programmed with 0x70, LED1 to LED4 are used for the backlight while LED5 to LED7 are individual sinks. Bit 7 in BLSEN is not used.

In backlight operation, there are three distinct selectable brightness levels: dark, office, and daylight. Each brightness level has two programmable settings:

- The MAX brightness defined in Register BLMX1 (daylight), Register BLMX2 (office), and Register BLMX3 (dark) can range from 0 mA to 30 mA.
- The DIM brightness level defined in Register BLDM1 for daylight, Register BLDM2 for office, and Register BLDM3 for dark can range from 0 mA to 30 mA.

The backlight LED current depends on the backlight transfer law, linear or square, programmed in the CFGR register Bit 2 and Bit 1. Table 2 shows all the current values available according to the control law selected.

The transfer function for the linear law is

$$BACKLIGHT\_I = RegValue \times \left(\frac{FULL\_SCALE\_I}{127}\right) \quad (1)$$

$$RegValue = \frac{BACKLIGHT\_I \times 127}{FULL\_SCALE\_I} \quad (2)$$

where:
BACKLIGHT_I is the desired LED current in mA.
RegValue is the digital 7-bit value programmed in the backlight current registers.
FULL_SCALE_I is the maximum backlight current value, which is 30 mA.

The transfer function for the square law is

$$BACKLIGHT\_I = \left(\frac{RegValue \times \sqrt{FULL\_SCALE\_I}}{127}\right)^2 \quad (3)$$

$$RegValue = \frac{127 \times \sqrt{BACKLIGHT\_I}}{\sqrt{FULL\_SCALE\_I}} \quad (4)$$

Table 1. Backlight Transfer Laws—CFGR Register Bits[2:1]

Bits[2:1]	Fading Law	Fading Time Change
00	Linear Law DAC	Linear time steps
01	Square Law DAC	Linear time steps
10	Square Law DAC (Cubic 1)	Nonlinear time steps (Type 1)
11	Square Law DAC (Cubic 2)	Nonlinear time steps (Type 2)

The complete set of square law current values are given in Table 2.

Table 2. Linear and Square Law Current Values

DAC Code	Linear Law (mA)	Square Law (mA)
0x00	0	0.000
0x01	0.236	0.002
0x02	0.472	0.007
0x03	0.709	0.017
0x04	0.945	0.030
0x05	1.181	0.047
0x06	1.417	0.067
0x07	1.654	0.091
0x08	1.890	0.119
0x09	2.126	0.151
0x0A	2.362	0.186
0x0B	2.598	0.225
0x0C	2.835	0.268
0x0D	3.071	0.314
0x0E	3.307	0.365
0x0F	3.543	0.419
0x10	3.780	0.476
0x11	4.016	0.538
0x12	4.252	0.603
0x13	4.488	0.671
0x14	4.724	0.744
0x15	4.961	0.820
0x16	5.197	0.900
0x17	5.433	0.984
0x18	5.669	1.071
0x19	5.906	1.163
0x1A	6.142	1.257
0x1B	6.378	1.356
0x1C	6.614	1.458
0x1D	6.850	1.564
0x1E	7.087	1.674
0x1F	7.323	1.787
0x20	7.559	1.905
0x21	7.795	2.026
0x22	8.031	2.150
0x23	8.268	2.279
0x24	8.504	2.411
0x25	8.740	2.546
0x26	8.976	2.686
0x27	9.213	2.829
0x28	9.449	2.976
0x29	9.685	3.127
0x2A	9.921	3.281
0x2B	10.157	3.439
0x2C	10.394	3.601
0x2D	10.630	3.767
0x2E	10.866	3.936
0x2F	11.102	4.109

DAC Code	Linear Law (mA)	Square Law (mA)
0x30	11.339	4.285
0x31	11.575	4.466
0x32	11.811	4.650
0x33	12.047	4.838
0x34	12.283	5.029
0x35	12.520	5.225
0x36	12.756	5.424
0x37	12.992	5.627
0x38	13.228	5.833
0x39	13.465	6.043
0x3A	13.701	6.257
0x3B	13.937	6.475
0x3C	14.173	6.696
0x3D	14.409	6.921
0x3E	14.646	7.150
0x3F	14.882	7.382
0x40	15.118	7.619
0x41	15.354	7.859
0x42	15.591	8.102
0x43	15.827	8.350
0x44	16.063	8.601
0x45	16.299	8.855
0x46	16.535	9.114
0x47	16.772	9.376
0x48	17.008	9.642
0x49	17.244	9.912
0x4A	17.480	10.185
0x4B	17.717	10.463
0x4C	17.953	10.743
0x4D	18.189	11.028
0x4E	18.425	11.316
0x4F	18.661	11.608
0x50	18.898	11.904
0x51	19.134	12.203
0x52	19.370	12.507
0x53	19.606	12.814
0x54	19.842	13.124
0x55	20.079	13.439
0x56	20.315	13.757
0x57	20.551	14.078
0x58	20.787	14.404
0x59	21.024	14.733
0x5A	21.260	15.066
0x5B	21.496	15.403
0x5C	21.732	15.743
0x5D	21.968	16.087
0x5E	22.205	16.435
0x5F	22.441	16.787

DAC Code	Linear Law (mA)	Square Law (mA)
0x60	22.677	17.142
0x61	22.913	17.501
0x62	23.150	17.863
0x63	23.386	18.230
0x64	23.622	18.600
0x65	23.858	18.974
0x66	24.094	19.351
0x67	24.331	19.733
0x68	24.567	20.118
0x69	24.803	20.507
0x6A	25.039	20.899
0x6B	25.276	21.295
0x6C	25.512	21.695
0x6D	25.748	22.099
0x6E	25.984	22.506
0x6F	26.220	22.917

DAC Code	Linear Law (mA)	Square Law (mA)
0x70	26.457	23.332
0x71	26.693	23.750
0x72	26.929	24.173
0x73	27.165	24.599
0x74	27.402	25.028
0x75	27.638	25.462
0x76	27.874	25.899
0x77	28.110	26.340
0x78	28.346	26.784
0x79	28.583	27.232
0x7A	28.819	27.684
0x7B	29.055	28.140
0x7C	29.291	28.599
0x7D	29.528	29.063
0x7E	29.764	29.529
0x7F	30.000	30.000

If the automatic light sensing control is enabled (CMP_AUTOEN, Bit 1, is set to 1 in Register MDCR) the result from the light sensing comparator controls the backlight brightness in one of the three levels (dark, office, and daylight).

In this mode, it is not possible to force, by software, a defined level. Thus, settings on Bits[4:3] (BLV) in the CFGR register are not considered in automatic light sensing mode. In case the automatic light sensing mode is disabled (0 is written to CMP_AUTOEN in the MDCR register), software can force the backlight brightness in one of the three possible modes (dark, office, or daylight) by writing Bits[4:3] (BLV) in Register CFGR (see Table 3).

Table 3. Brightness Level—Bits[4:3] (BLV) in CFGR Register

Bits[4:3]	Backlight Brightness Level
00	Level 1 (daylight)
01	Level 2 (office)
10	Level 3 (dark)
11	Disabled

Each individual sink has a dedicated current register defining its brightness level. As with the backlight, the current level depends on the transfer law selected in the ISCFR register Bits[1:0] (SC_LAW).

The ISC1 register defines the current level for LED1, the ISC2 register defines the level for LED2, and so on until ISC7, which controls the current for LED7. The maximum current level programmable is 30 mA. However, LED7 can be set up to 60 mA if Bit 7 in the ISC7 register is set to 1. The possible current levels for 30 mA are listed in Table 2.

Table 4. Complete Set of Square Law Current Values for LED7—60 mA Range

DAC Code	Linear Law (mA)	Square Law (mA)
0x00	0.000	0
0x01	0.472	0.004
0x02	0.945	0.014
0x03	1.42	0.034
0x04	1.89	0.06
0x05	2.36	0.094
0x06	2.83	0.134
0x07	3.31	0.182
0x08	3.78	0.238
0x09	4.25	0.302
0x0A	4.72	0.372
0x0B	5.20	0.45
0x0C	5.67	0.536
0x0D	6.14	0.628
0x0E	6.61	0.73
0x0F	7.09	0.838
0x10	7.56	0.952
0x11	8.03	1.076
0x12	8.50	1.206
0x13	8.98	1.342
0x14	9.45	1.488
0x15	9.92	1.64
0x16	10.39	1.8
0x17	10.87	1.968
0x18	11.34	2.142
0x19	11.81	2.326
0x1A	12.28	2.514
0x1B	12.76	2.712
0x1C	13.23	2.916
0x1D	13.70	3.128
0x1E	14.17	3.348
0x1F	14.65	3.574
0x20	15.12	3.81
0x21	15.59	4.052
0x22	16.06	4.3
0x23	16.54	4.558
0x24	17.01	4.822
0x25	17.48	5.092
0x26	17.95	5.372
0x27	18.43	5.658
0x28	18.90	5.952
0x29	19.37	6.254
0x2A	19.84	6.562
0x2B	20.31	6.878
0x2C	20.79	7.202
0x2D	21.26	7.534
0x2E	21.73	7.872
0x2F	22.20	8.218
0x30	22.68	8.57
0x31	23.15	8.932
0x32	23.62	9.3
0x33	24.09	9.676
0x34	24.57	10.058
0x35	25.04	10.45
0x36	25.51	10.848
0x37	25.98	11.254
0x38	26.46	11.666
0x39	26.93	12.086
0x3A	27.40	12.514
0x3B	27.87	12.95
0x3C	28.35	13.392
0x3D	28.82	13.842
0x3E	29.29	14.3
0x3F	29.76	14.764
0x40	30.24	15.238
0x41	30.71	15.718
0x42	31.18	16.204
0x43	31.65	16.7
0x44	32.13	17.202
0x45	32.60	17.71
0x46	33.07	18.228
0x47	33.54	18.752
0x48	34.02	19.284
0x49	34.49	19.824
0x4A	34.96	20.37
0x4B	35.43	20.926
0x4C	35.91	21.486
0x4D	36.38	22.056
0x4E	36.85	22.632
0x4F	37.32	23.216
0x50	37.80	23.808
0x51	38.27	24.406
0x52	38.74	25.014
0x53	39.21	25.628
0x54	39.69	26.248
0x55	40.16	26.878
0x56	40.63	27.514
0x57	41.10	28.156
0x58	41.57	28.808
0x59	42.05	29.466
0x5A	42.52	30.132
0x5B	42.99	30.806

DAC Code	Linear Law (mA)	Square Law (mA)
0x5C	43.46	31.486
0x5D	43.94	32.174
0x5E	44.41	32.87
0x5F	44.88	33.574
0x60	45.35	34.284
0x61	45.83	35.002
0x62	46.30	35.726
0x63	46.77	36.46
0x64	47.24	37.2
0x65	47.72	37.948
0x66	48.19	38.702
0x67	48.66	39.466
0x68	49.13	40.236
0x69	49.61	41.014
0x6A	50.08	41.798
0x6B	50.55	42.59
0x6C	51.02	43.39
0x6D	51.50	44.198

DAC Code	Linear Law (mA)	Square Law (mA)
0x6E	51.97	45.012
0x6F	52.44	45.834
0x70	52.91	46.664
0x71	53.39	47.5
0x72	53.86	48.346
0x73	54.33	49.198
0x74	54.80	50.056
0x75	55.28	50.924
0x76	55.75	51.798
0x77	56.22	52.68
0x78	56.69	53.568
0x79	57.17	54.464
0x7A	57.64	55.368
0x7B	58.11	56.28
0x7C	58.58	57.198
0x7D	59.06	58.126
0x7E	59.53	59.058
0x7F	60	60

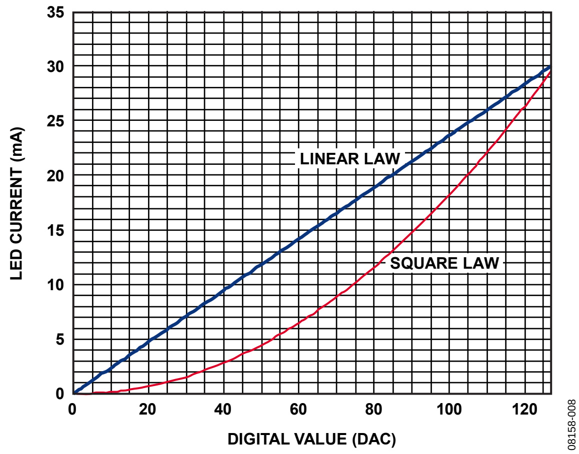

Figure 8. LED Current Transfer Law Curves

Square Cubic Law 1 and Square Cubic Law 2 use the same square transfer law values with faster increment/decrement times when fading in and out.

FADE OVERRIDE MODE

The fade override mode is enabled when Bit 0 (FOVR) in the CFGR register is set to 1. In this mode, the backlight skips the fade-in time when Bit BLEN in the MDCR register is set to 1 and it goes directly to the programmed maximum backlight brightness intensity. There remains a 100 ms ramp-up time to avoid rapid change in current. The fade-in time is not skipped at the very first activation, after an off condition.

This mode is useful is situations where the backlight is fading out and the user depresses a keypad button. The system software can intercept this condition, set the FOVR bit, then write the BLEN bit to 1 again. Fade-out time is not affected by the FOVR bit.

SETTING BACKLIGHT AND SINK TIMINGS

The LED backlight supports several programmable timing controls as described in this section.

CONTROLLING BACKLIGHT SETTINGS

Bits[3:0] of the BLFR register set the backlight fade-in time (see Table 5). The time specified in Table 5 is from the maximum allowable current level (30 mA) to zero. Equation 5 gives an estimated value for the actual fade-in time based on known conditions. If the value programmed is zero, the fade-in time is disabled, however a 100 ms fade-in time is used.

Bits[7:4] of the BLFR register set the backlight fade-out time, (see Table 5). Equation 6 gives an estimated value for the actual fade-in time based on known conditions. If the value programmed is zero, the fade-out time is disabled, however a 100 ms fade-in time is used.

Bits[6:0] of the BLDIM register set the time for the backlight to stay in the MAX brightness state. After this time, the backlight starts to fade out to the DIM or zero level (If the DIM timeout is disabled). The digital value programmed in this register corresponds to the time in seconds. For example, BLDIM = 0x32 corresponds to 50 seconds.

Bits[6:0] of the BLOFF register set the time for the backlight to stay in the DIM state. After this time, the backlight starts to fade out to zero current level. The digital value programmed in this register corresponds to the time in seconds. For example, BLOFF = 0x10 corresponds to 16 seconds.

Table 5. Backlight Fade-In (Bits[3:0]) and Fade-Out (Bits[7:4]) Times

BLFR Register Value	Fade-In and Fade-Out Time (Sec)
0000	Disabled
0001	0.3
0010	0.6
0011	0.9
0100	1.2
0101	1.5
0110	1.8
0111	2.1
1000	2.4
1001	2.7
1010	3.0
1011	3.5
1100	4.0
1101	4.5
1110	5.0
1111	5.5

Note that the following equation is only for linear step control laws:

$$T_{FADE\_EST} = \frac{Fade\_In \times (MaxCurr - ActualCurr)}{30} \quad (5)$$

where:
MaxCurr is the maximum backlight current value programmed in the BLMXx registers.
ActualCurr is the initial backlight current (0 mA at the first activation).
T_{FADE_EST} is the calculated fade-in time in seconds.

Note that the following equation is only for linear step control laws:

$$T_{FADE\_EST} = \frac{Fade\_Out \times (ActualCurr - MinCurr)}{30} \quad (6)$$

where:
ActualCurr is the backlight current value and can be MaxCurr if fading out from the MAX brightness state or DimCurr if fading out from DIM state.
MinCurr is the backlight current value and can be DimCurr if fading out from the MAX brightness to DIM state or zero if fading out from DIM state to zero current.
T_{FADE_EST} is the calculated fade-out time in seconds.

CONTROLLING SINK SETTINGS

The enabled individual sinks have dedicated registers to control the activation timings and fade-in, fade-out times. The activation on time for Sink 1 to Sink 7 is selected by Register ISCT1, Bits[7:6] and is common for all the individual sinks (see Table 6).

The off time is selected individually for each individual sink (Sink 1 to Sink 7) by Register ISCT1, Bits[5:0] and Register ISCT2, Bits[7:0] (see Table 7). Two bits are used per individual sink. If the off time, 2-bit register is set to zero, the individual sink is activated indefinitely.

The off time is controlled on each individual sink as follows: ISCT1 Bits[5:4] for Sink 7, ISCT1 Bits[3:2] for Sink 6, ISCT1 Bits[1:0] for Sink 5, ISCT2 Bits[7:6] for Sink 4, ISCT2 Bits[5:4] for Sink 3, ISCT2 Bits[3:2] for Sink 2 and ISCT2 Bits[1:0] for Sink 1.

Table 6. Individual Sinks On-Time Selection

ISCT1 Bits[7:6]	Programmed On Time (Sec)
00	0.2
01	0.6
10	0.8
11	1.2

Table 7. Individual Sinks Off-Time Selection

Two-Bit Register	Programmed Off Time (Sec)
00	Always on
01	0.6
10	1.2
11	1.8

Bits[3:0] of the ISCF register set the individual sinks fade-in time (see Table 8). The time specified in Table 8 is from zero to the maximum allowable current level (30 mA or 60 mA for Sink 7). Equation 5 gives an estimated value for the fade-in time based on the actual programmed LED current. If the value programmed is zero, the fade-in time is disabled; however, a 100 ms fade-in time is used.

Bits[7:4] of the ISCF register set the individual sinks fade-out time (see Table 8). The time specified in Table 8 is from the maximum allowable current (30 mA or 60 mA for Sink 7) to zero current. Equation 6 gives an estimated value for the fade-out time based on actual programmed LED current. If the value programmed is zero, the fade-out time is disabled; however, a 100 ms fade-in time is used.

Table 8. Individual Sinks Fade-In and Fade-Out Times

Register Value	Fade-In and Fade-Out Time in Seconds
0000	Disabled
0001	0.3
0010	0.6
0011	0.9
0100	1.2
0101	1.5
0110	1.8
0111	2.1
1000	2.4
1001	2.7
1010	3.0
1011	3.5
1100	4.0
1101	4.5
1110	5.0
1111	5.5

LIGHT SENSING COMPARATOR

The ADP8860 supports two light sensing inputs. The first light sensing input is always dedicated as input from a phototransistor assessing the ambient light condition. The second light sensing input is shared with Sink 6. Thus, the software must carefully select the desired operation (light sensing or Sink 6).

Bit 5 (CMP2_SEL) in the CFGR register is used to configure pin D6 (LED6) operation. If CMP2_SEL is 0, Pin D6 is an output sink for LED6. If CMP2_SEL is set to 1, Pin D6 is an input from the second light sensor (phototransistor). An internal analog-to-digital converter processes the light information coming from the light sensors providing a digital value corresponding to the ambient light detected. The digital value is compared to two thresholds (L2 and L3) to determine if the ambient light condition is in one of the following three ranges:

- Dark: Ambient light level is below L3 and L2 thresholds.
- Office: Ambient light level is above L3, but below L2 thresholds.
- Daylight: Ambient light level is above L3 and L2 thresholds.

Software can program the ADP8860 to operate in automatic mode if Bit 1 (CMP_AUTOEN) in the MDCR register is set to 1 or in manual mode, if CMP_AUTOEN is 0. When the device operates in automatic light ambient mode, the backlight brightness level is set by the current ambient light condition (dark, office, and daylight).

Bit 0 and Bit 1 in CCFG and CCFG2 allow selection of the level (L2 and/or L3) used for the automatic brightness control. If both bits are 0, no automatic control is issued (see Table 9). Software must program the desired levels for the L2 and L3 thresholds in the L2_TRP and L3_TRP registers, respectively. Using the default phototransistor, the correlation factor between programmed value and light level is provided in Table 10.

Table 9. Comparators Auto Level Control

Code	Comparator Operation
00	No automatic brightness control
01	L2 comparator auto level control
10	L3 comparator auto level control
11	L2 and L3 auto level control

Table 10. Correlation Between Code and Phototransistor Lux Measurement

Threshold	Registers	1-Bit Correlation
L2	L2_TRP, L2_HYS	10 lux or 4 μA
L3	L3_TRP, L3_HYS	1.25 lux or 0.5 μA

Each threshold has an associated register setting the hysteresis level to avoid changes due to noise or being too close to the threshold. Each light sensor has an individual comparator and control register.

Table 11 shows the filter settings for the comparators. The comparator filter makes sure that rapid changes in the ambient light level do not change the backlight brightness.

Table 11. Comparators Filter Settings—Bits[7:5] in CCFG and CCFG2

Code	Filter Setting (ms)
000	80
001	160
010	320
011	640
100	1280
101	2560
110	5120
111	10240

Every time the L2 or L3 threshold is crossed (rising or falling conditions), a pending flag is set in the interrupt pending register (MDCR2). Bit 0 (CMP_INT) is the flag for the first light sensor while Bit 1 (CMP2_INT) is the flag for the second light sensor. An interrupt is generated if the respective Interrupt Enable bit is 1 in the INTR_EN register (or the CMP_IEN register for light Sensor 1 and the CMP2_IEN register for light Sensor 2).

The comparator interrupt can be useful in case the manual backlight brightness level mode of operation is used so that the software can read the comparator(s) outputs and force the backlight brightness level through Bit BLV in the CFGR register. See Table 12 for the complete comparator output decoding.

The PH1LEVL, PH1LEVH, PH2LEVL, and PH2LEVH registers contain the raw 13-bit conversion result for the first and second light sensors. These registers can be used for diagnostic or monitoring purposes. In typical applications, these registers can be ignored.

Table 12. Light Sensor Comparator Levels

L2_OUT(2)	L3_OUT(2)	Ambient Light Level	Condition
0	0	Ambient light > L2	Daylight
1	0	L3< ambient light < L2	Office
1	1	Ambient light < L3	Dark

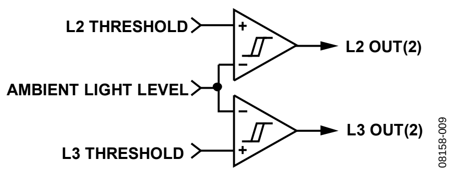

Figure 9. Light Sensing Comparators

The ambient light comparators have hysteresis thresholds (L2_HYS and L3_HYS) added to the respective tripping registers (L2_TRP and L3_TRP) to avoid the backlight mode changes because of noise.

Figure 10 shows how the backlight mode of operation is changed when the ambient light intensity varies.

Note that when the automatic light control mode and the backlight are activated the first time, the state machine converts the information coming from the phototransistor immediately (takes roughly 82 ms to perform the conversion). The comparator result is then used to set the backlight intensity without waiting for the comparator filter result. This avoids abrupt brightness changes due to the filter delay calculating a new value.

When the comparator is programmed to control the backlight intensity (setting Bit CMP_AUTOEN in the MDCR register), Bits[4:3] (BLV) in the CFGR register are modified with the backlight operating state. Table 13 shows the decoding for CFGR Bits[4:3] and CMP_AUTOEN Bit 1. When using this table, note that R/W indicates a read/write register and R indicates a read only register.

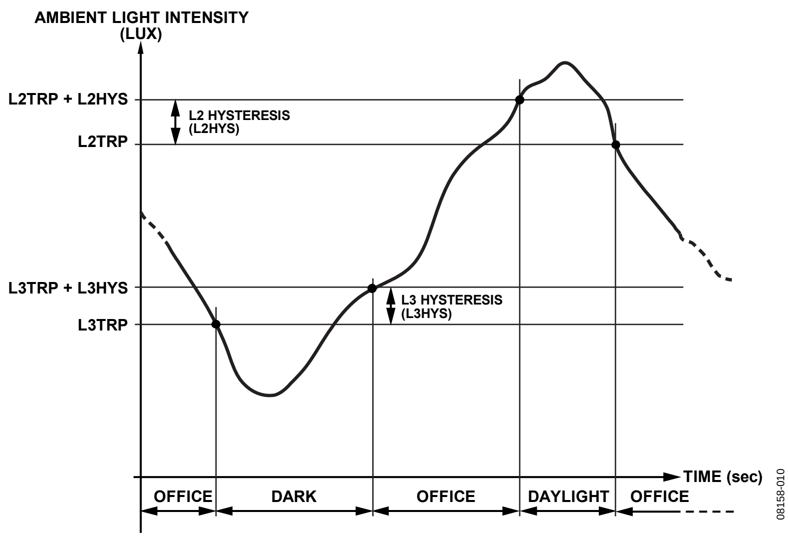

Figure 10. Comparator Thresholds

Table 13. Backlight Forced Levels

CFGR[4]	CFGR[3]	MDCR[1]	Description
0 (R/W)	0 (R/W)	0	Set daylight brightness level
0 (R/W)	1 (R/W)	0	Set office brightness level
1 (R/W)	0 (R/W)	0	Set dark brightness level
1 (R/W)	1 (R/W)	0	Disabled
0 (R)	0 (R)	1	Comparator forces daylight level
0 (R)	1 (R)	1	Comparator forces office level
1 (R)	0 (R)	1	Comparator forces level
1 (R)	1 (R)	1	Comparator not enabled (Lx_EN = 0)

CONVENTIONS

Bit registers are defined as:

register_name.bit_name

For example

MDCR.7 = 1

sets Bit 7 in Register MDCR to 1.

In some cases, more than one bit is used to define a subregister function. For example, two bits define the LAW in the CFGR register. To present the data structure in a simpler format, the C language structure has been adopted. For example, the structure presented in this user guide defines the subregister field and dimension.

Note the following regarding this code:

- A colon is used to define a range of bits in one specific register. For example, BLFR[7:4] specifies the upper four bits in the BLFR register.
- In general, variables appear in lowercase while constants appear in uppercase.
- Hexadecimal numbers are specified with the 0x prefix. For example, 0x40 corresponds to 64 in decimal notation.
- Decimal numbers are the default notation and can also be specified with a postfix of dec, as in 120dec.
- Binary numbers are specified with a postfix of b as in 00110101b. This corresponds to the hexadecimal number 0x35.

```
/*****************************************************
#define _CFGR 0x04               /* This is the Configuration Register */
struct reg_0x04 {
      unsigned FOVR : 1;         /* This is Bit 0 */c
      unsigned LAW : 2;          /* This is Bit 1 and Bit 2 */
      unsigned BLV : 2;          /* This is Bit 4 and Bit 3 */
      unsigned CMP2_SEL : 1;     /* This is Bit 5 */
      unsigned SEL_AB : 1;       /* This is Bit 6 */
      };
```

Now the register can be defined and used as follows:

```
struct reg_0x04 cfgr;     /* Associate cfgr variable with the specific structure */
cfgr.LAW = 2              /* Write LAW bits only */
```

FUNCTIONAL FLOWCHARTS

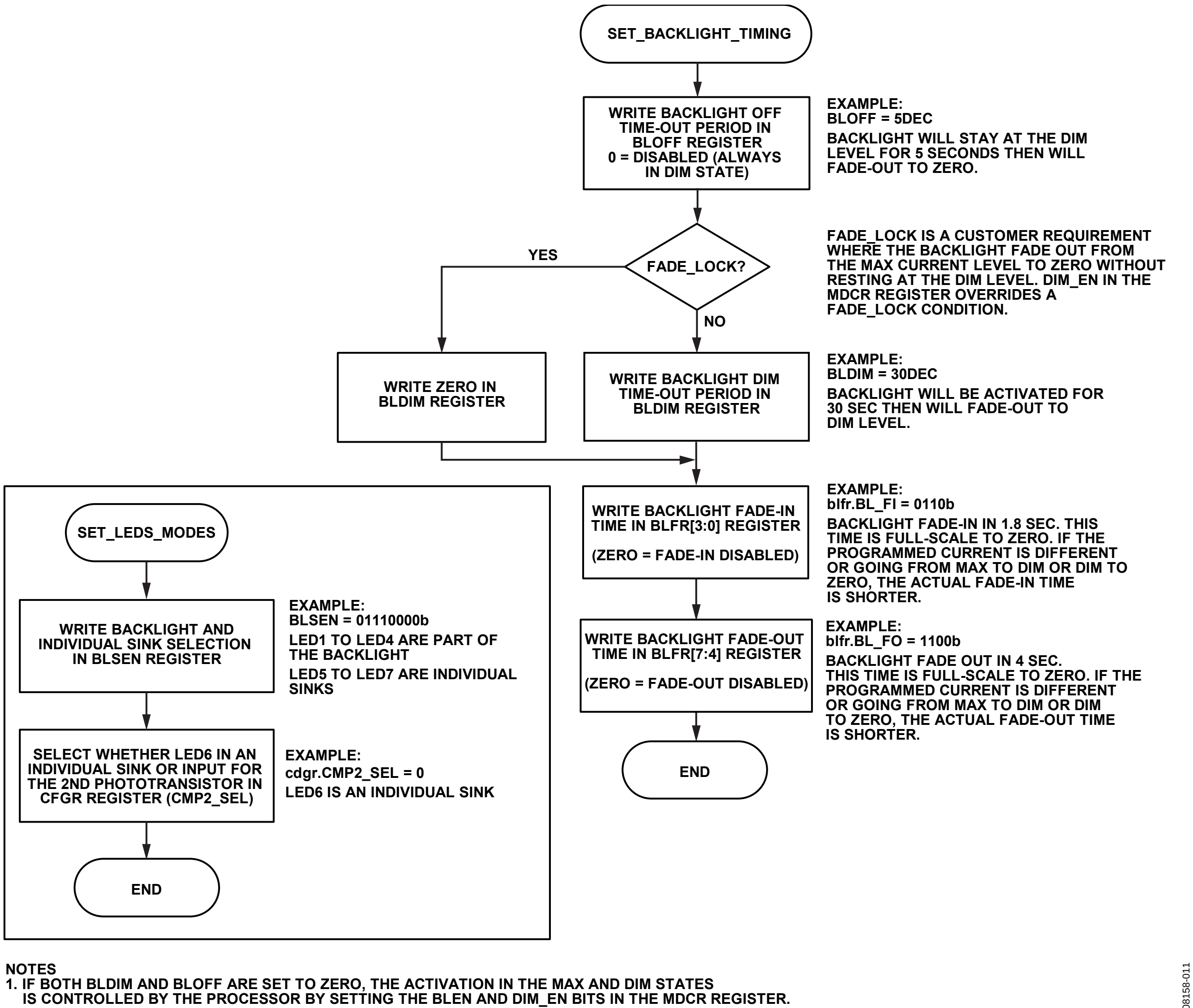

Figure 11. Set Backlight Timing Flowchart

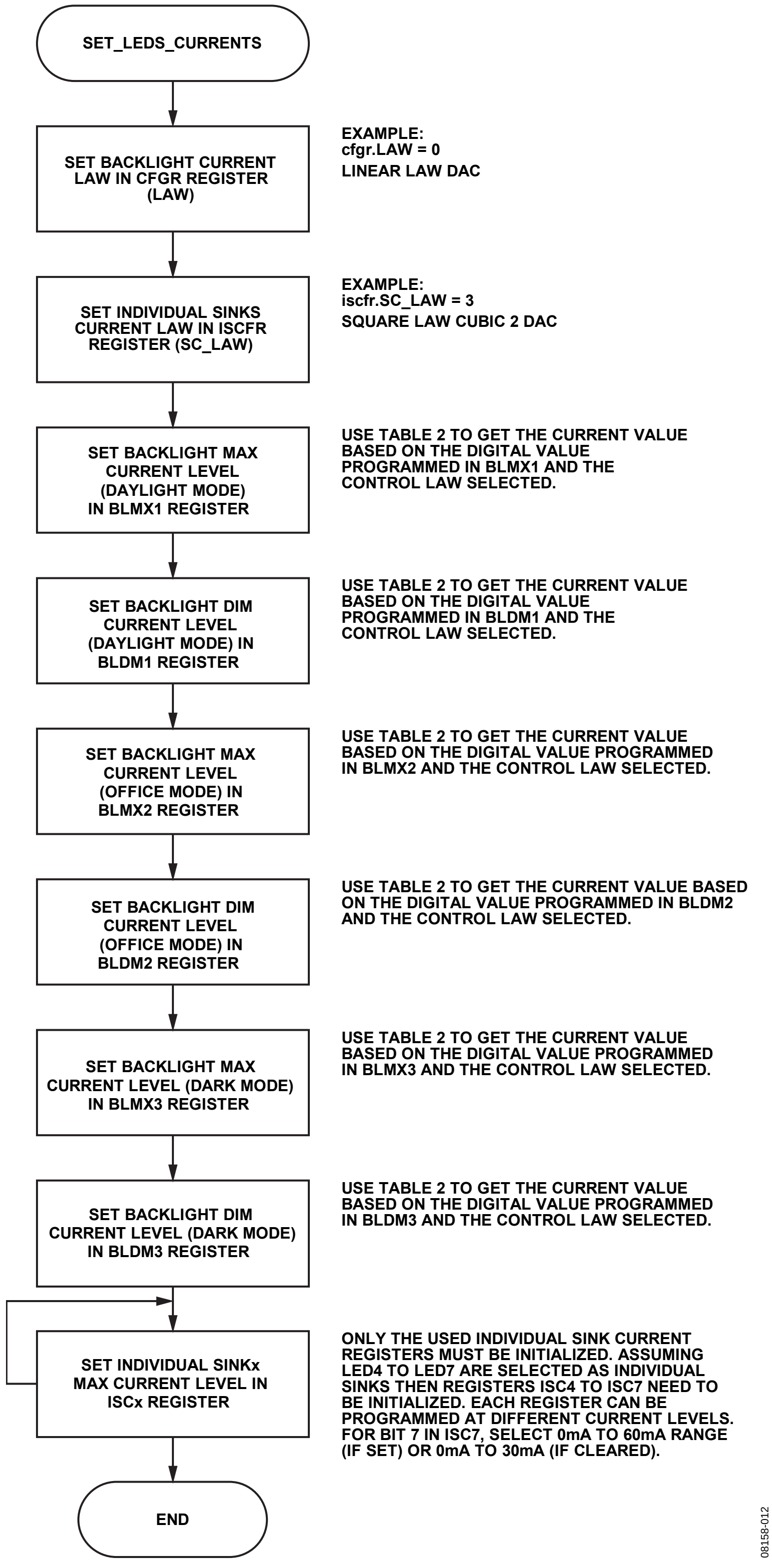

Figure 12. Set LEDs Currents Flowchart

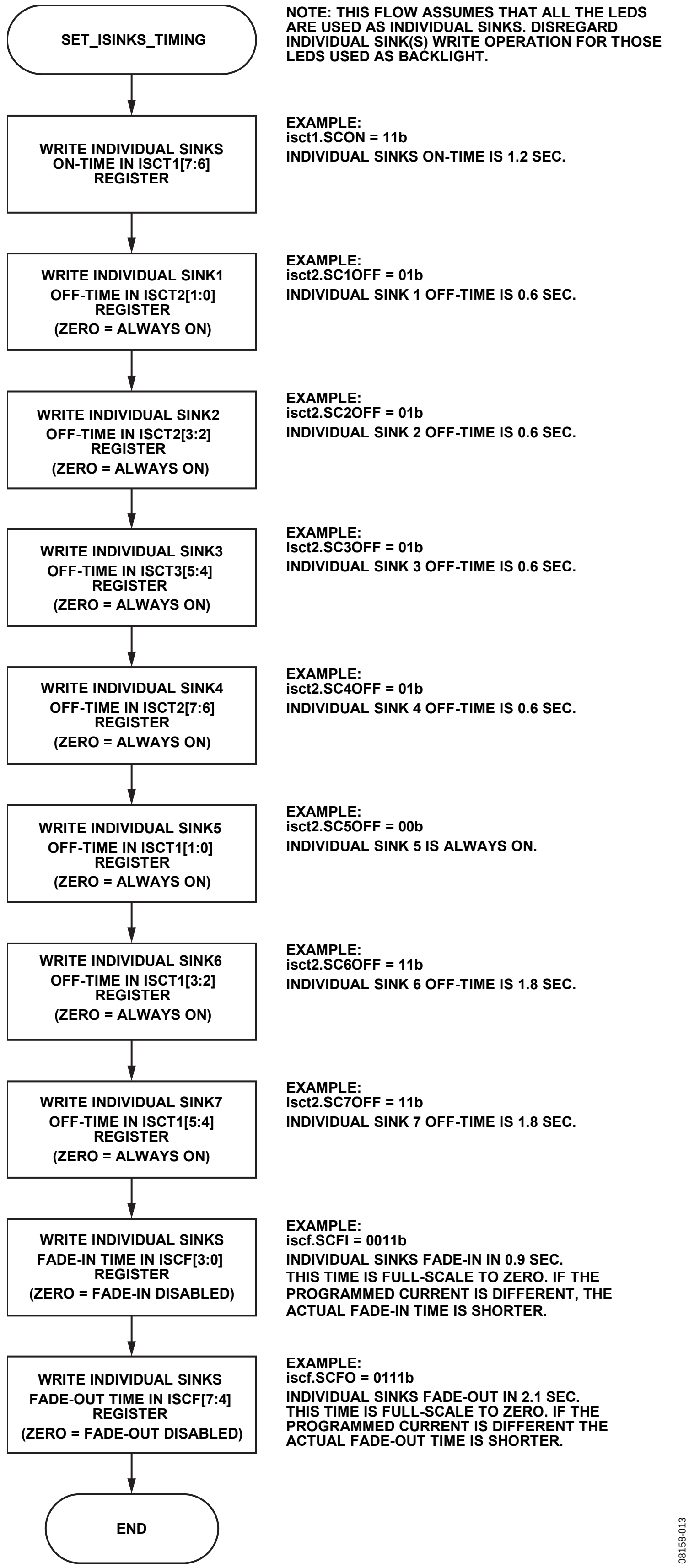

Figure 13. Set ISINKS Timing Flowchart

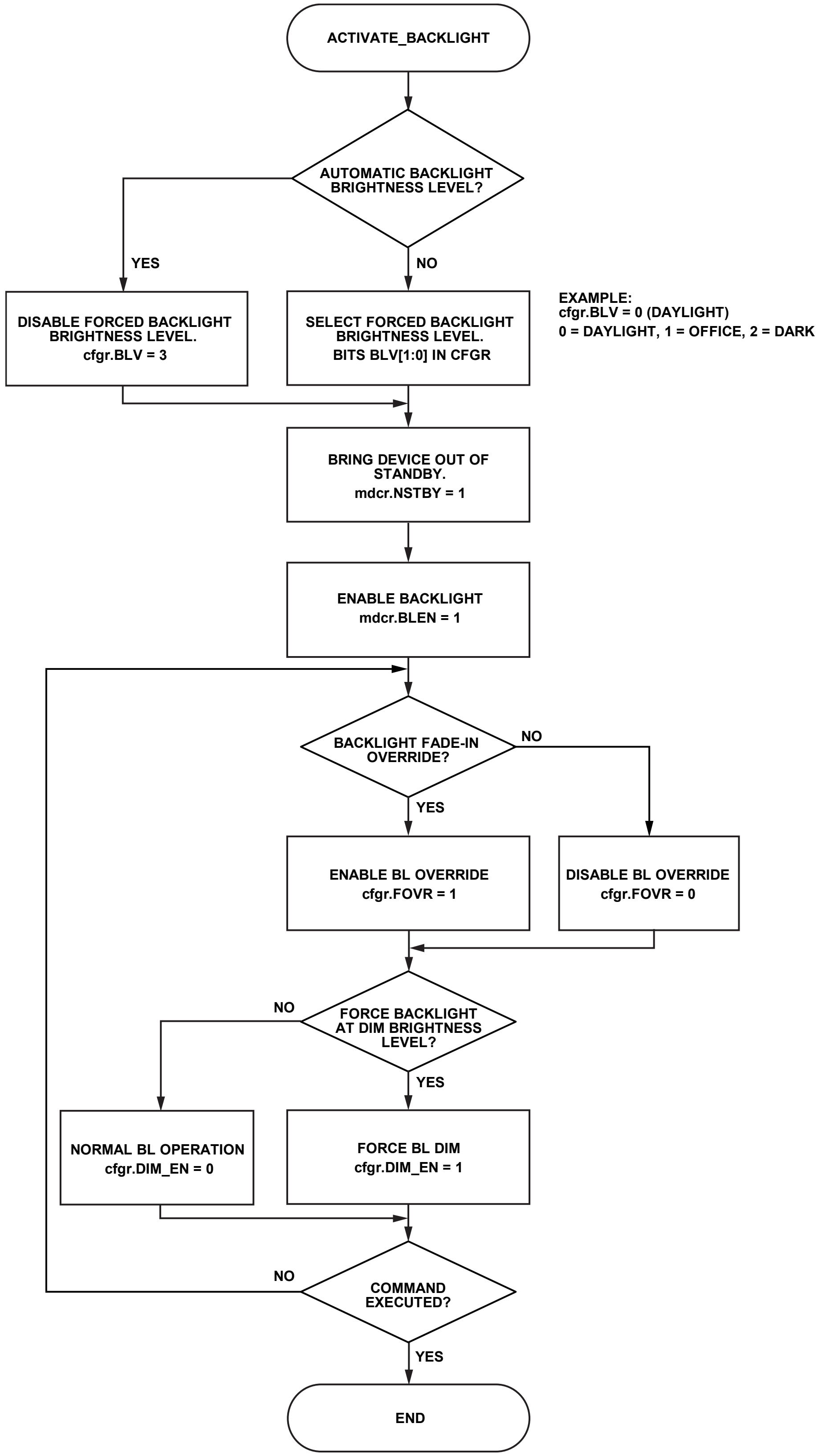

Figure 14. Activate Backlight Flowchart

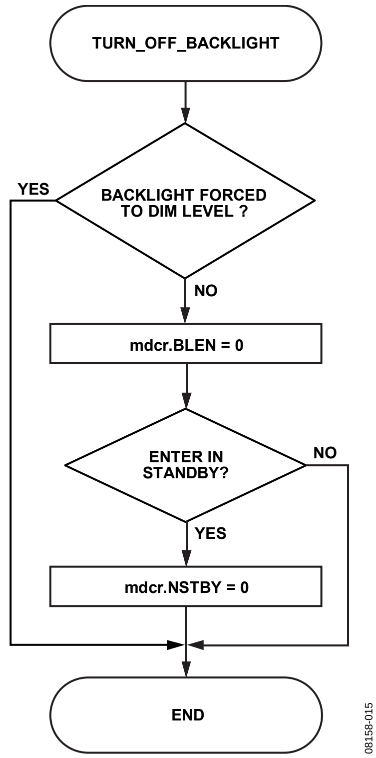

Figure 15. Turn Off Backlight Flowchart

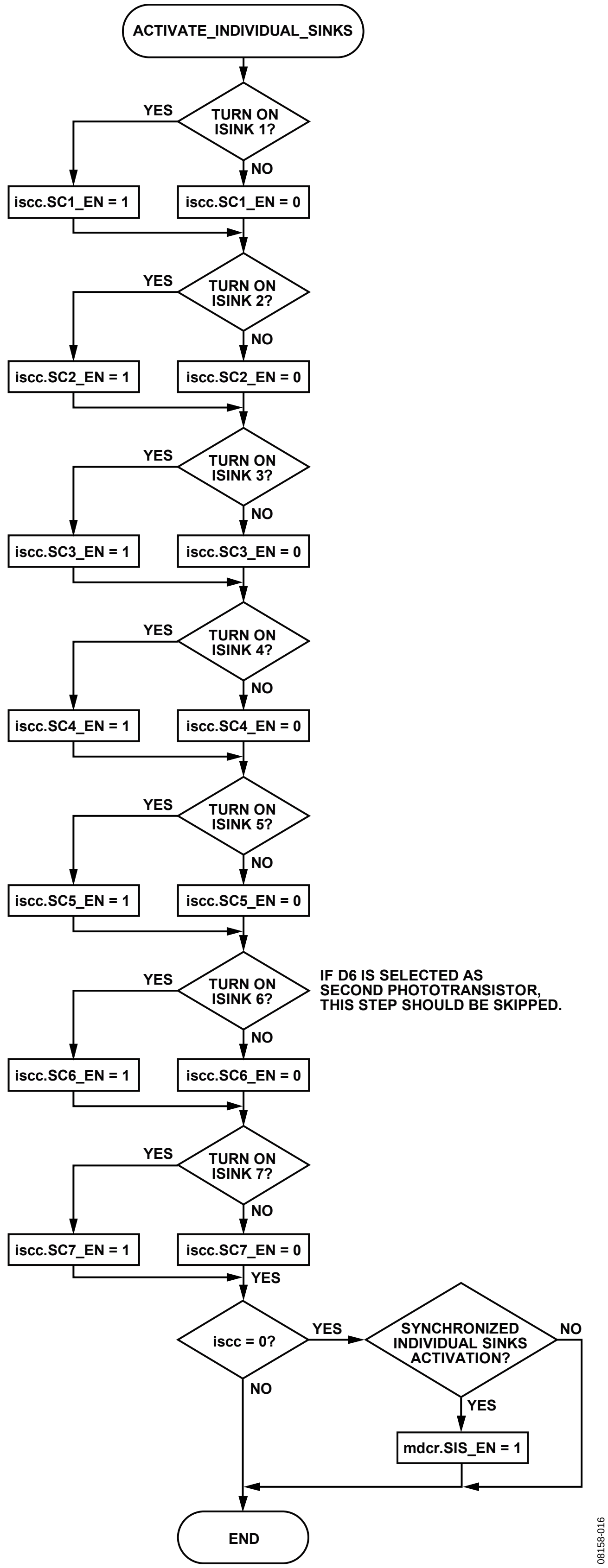

Figure 16. Activate Individual Sinks Flowchart

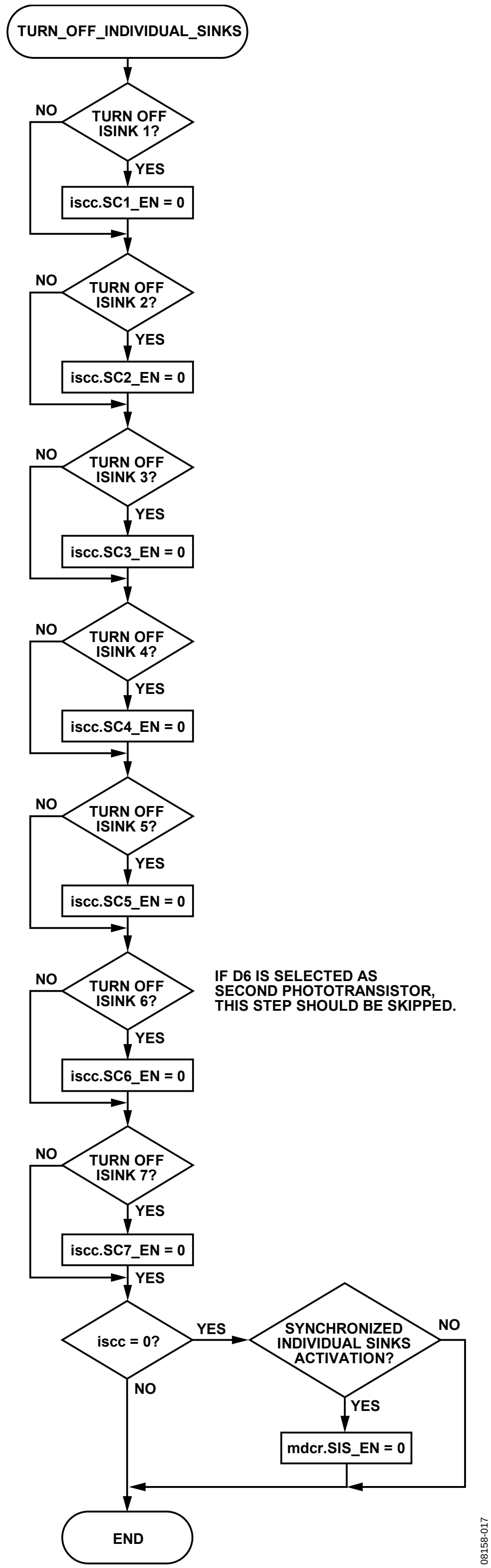

Figure 17. Turn Off Individual Sinks Flowchart

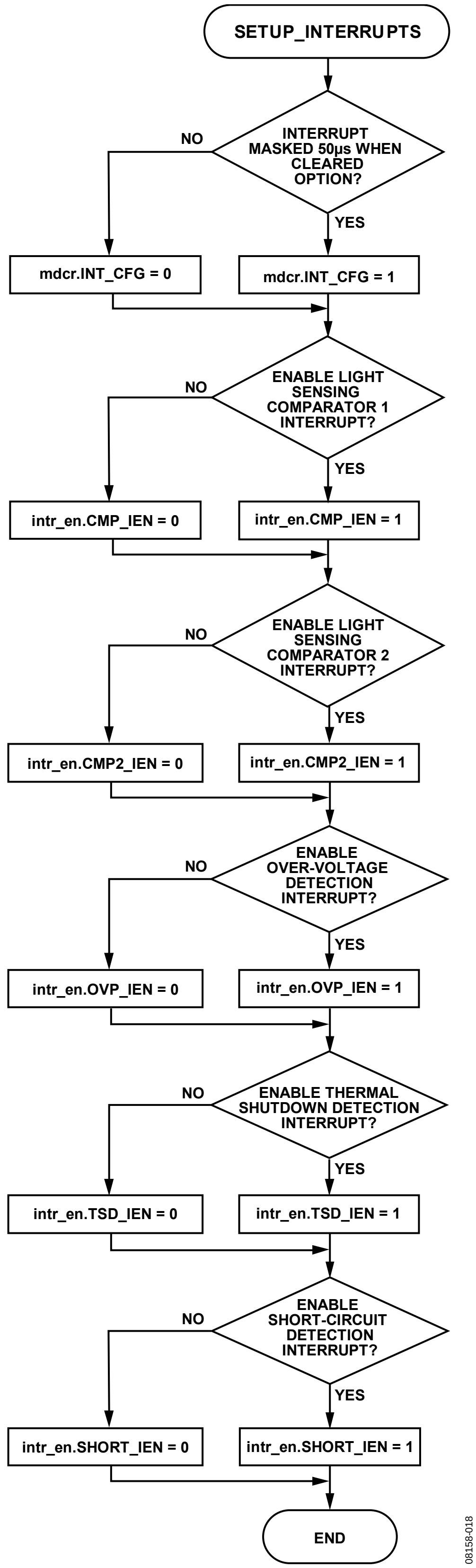

Figure 18. Setup Interrupts Flowchart

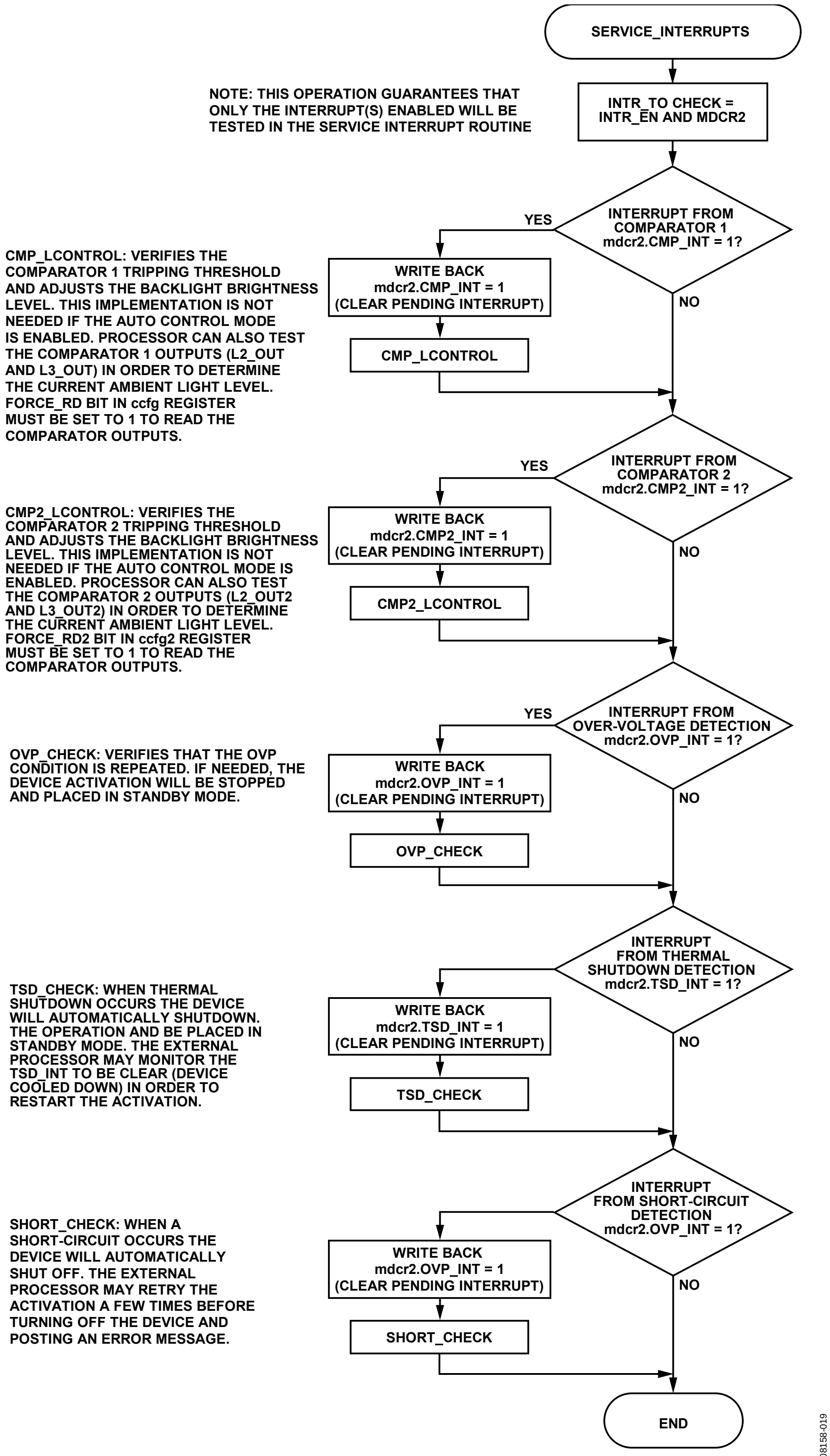

Figure 19. Service Interrupts Flowchart

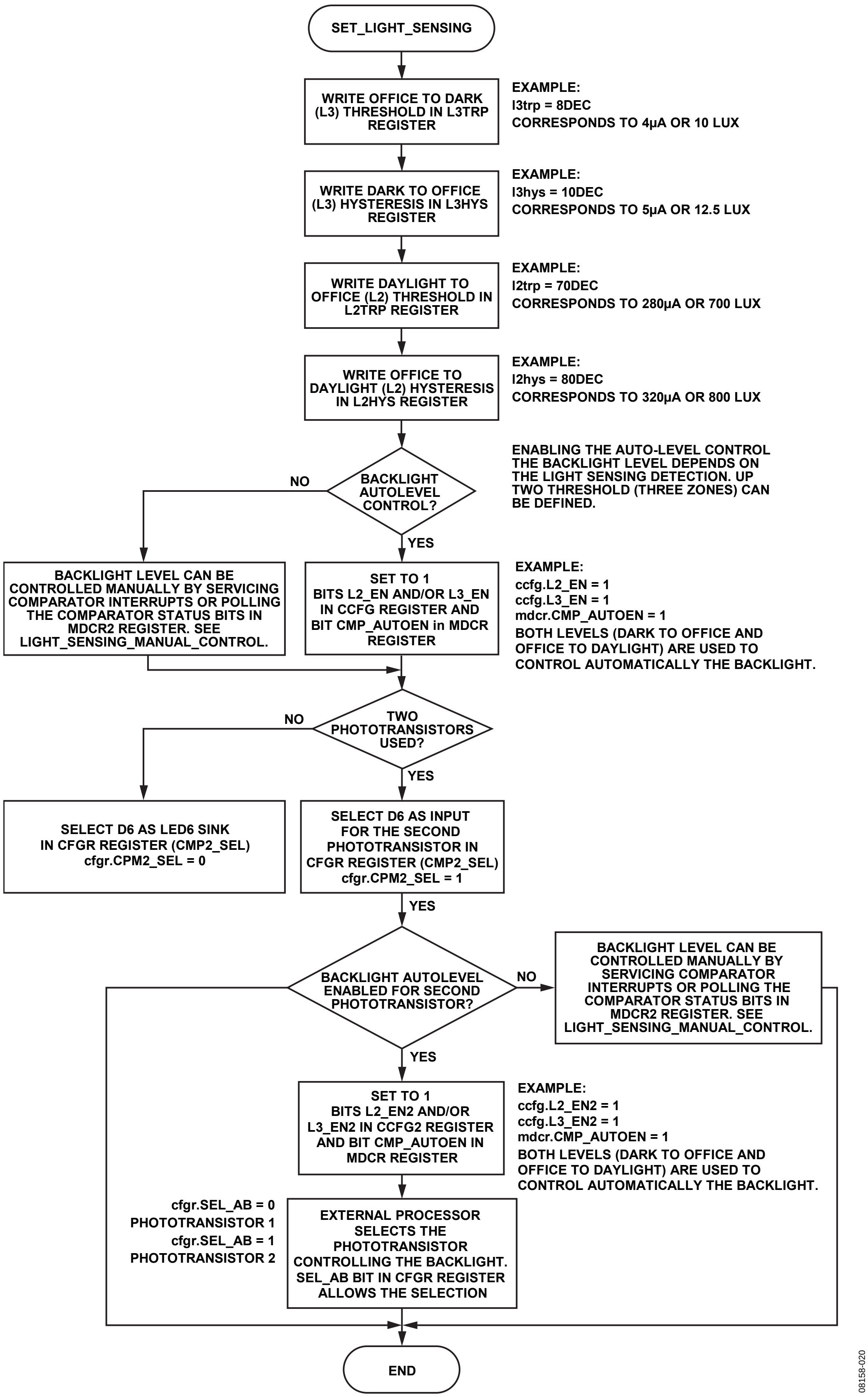

Figure 20. Set Light Sensing Flowchart

REGISTERS MAP

Table 14.

Addr	Reg. Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	MFDVID	Manufacture ID				Device ID			
0x01	MDCR	Reserved	INT_CFG	NSTBY	DIM_EN	Reserved	SIS_EN	CMP_AUTOEN	BLEN
0x02	MDCR2	Reserved			SHORT_INT	TSD_INT	OVP_INT	CMP2_INT	CMP_INT
0x03	INTR_EN	Reserved			SHORT_IEN	TSD_IEN	OVP_IEN	CMP2_IEN	CMP_IEN
0x04	CFGR	Reserved	SEL_AB	CMP2_SEL	BLV		Law		FOVR
0x05	BLSEN	Reserved	D7EN	D6EN	D5EN	D4EN	D3EN	D2EN	D1EN
0x06	BLOFF	Reserved	OFFT						
0x07	BLDIM	Reserved	DIMT						
0x08	BLFR	BL_FO				BL_FI			
0x09	BLMX1	Reserved	BL1_MC						
0x0A	BLDM1	Reserved	BL1_DC						
0x0B	BLMX2	Reserved	BL2_MC						
0x0C	BLDM2	Reserved	BL2_DC						
0x0D	BLMX3	Reserved	BL3_MC						
0x0E	BLDM3	Reserved	BL3_DC						
0x0F	ISCFR	Reserved						SC_LAW	
0x10	ISCC	Reserved	SC7_EN	SC6_EN	SC5_EN	SC4_EN	SC3_EN	SC2_EN	SC1_EN
0x11	ISCT1	SCON		SC7OFF		SC6OFF		SC5OFF	
0x12	ISCT2	SC4OFF		SC3OFF		SC2OFF		SC1OFF	
0x13	ISCF	SCFO				SCFI			
0x14	ISC7	SCR	SCD7						
0x15	ISC6	Reserved	SCD6						
0x16	ISC5	Reserved	SCD5						
0x17	ISC4	Reserved	SCD4						
0x18	ISC3	Reserved	SCD3						
0x19	ISC2	Reserved	SCD2						
0x1A	ISC1	Reserved	SCD1						
0x1B	CCFG	FILT			FORCE_RD	L3_OUT	L2_OUT	L3_EN	L2_EN
0x1C	CCFG2	FILT2			FORCE_RD2	L3_OUT2	L2_OUT2	L3_EN2	L2_EN2
0x1D	L2_TRP	L2_TRP							
0x1E	L2_HYS	L2_HYS							
0x1F	L3_TRP	L3_TRP							
0x20	L3_HYS	L3_HYS							
0x21	PH1LEVL	PH1LEV_LOW							
0x22	PH1LEVH	Reserved			PH1LEV_HIGH				
0x23	PH2LEVL	PH2LEV_LOW							
0x24	PH2LEVH	Reserved			PH2LEV_HIGH				

Refer to the ADP8860 data sheet for a detailed description of each register.

PSEUDOCODE IMPLEMENTATION

The pseudo language implementation presented in this user guide uses a C-like programming syntax to describe the registers and functions needed to operate the backlight, individual sinks, and automatic light sensing. The goal is to provide a clear understanding for the device functionality and this may not be the most optimized approach for the device driver development. For example, the registers are being implemented as a C structure to better visualize and label the bit mapping of each register; however, in a real application constants can be used since there is no need to have variables allocated for the ADP8860.

LISTING FOR ADP8860_REGS.H

```
/*********************************************************************************/
#define _MFDVID 0x00       /* Manufacturing and device ID address */
struct reg_0x00     {
       unsigned DEVICE_ID : 4;
       unsigned MANUFACTURER_ID : 4;
                           };
/*********************************************************************************/
#define _MDCR 0x01         /* Device control and status */
struct reg_0x01     {
       unsigned BLEN : 1;
       unsigned CMP_AUTOEN : 1;
       unsigned SIS_EN : 1;
       unsigned RESERVED : 1;
       unsigned DIM_EN : 1;
       unsigned NSTBY : 1;
                    };
/*********************************************************************************/
#define _MDCR2 0x02        /* Interrupt pending flags */
struct reg_0x02  {
       unsigned CMP_INT : 1;
       unsigned CMP2_INT : 1;
       unsigned OVP_INT : 1;
       unsigned TSD_INT : 1;
       unsigned SHORT_INT : 1;
                      };
/*********************************************************************************/
#define _INTR_EN 0x03      /* Interrupt enable bits */
struct reg_0x03  {
       unsigned CMP_IEN : 1;
       unsigned CMP2_IEN : 1;
       unsigned OVP_IEN : 1;
       unsigned TSD_IEN : 1;
       unsigned SHORT_IEN : 1;
                      };
```

```
/*****************************************************************************/
#define _CFGR 0x04         /* Configuration register */
struct reg_0x04 {
      unsigned FOVR : 1;
      unsigned LAW : 2;
      unsigned BLV : 2;
      unsigned CMP2_SEL : 1;
      unsigned SEL_AB : 1;
                          };
/*****************************************************************************/
#define _BLSEN 0x05        /* Backlight or individual sink selection */
struct       reg_0x05      {
      unsigned D1EN : 1;
      unsigned D2EN : 1;
      unsigned D3EN : 1;
      unsigned D4EN : 1;
      unsigned D5EN : 1;
      unsigned D6EN : 1;
      unsigned D7EN : 1;
                          };
/*****************************************************************************/
#define _BLOFF 0x06        /* Backlight OFF timeout */
unsigned char offt = 0;    /* 0 = timeout disabled, value range 1 to 127, 1-bit = 1 sec. */
/*****************************************************************************/
#define _BLDIM 0x07        /* Backlight DIM timeout */
unsigned char dimt;        /* 0 = timeout disabled, value range 1 to 127, 1-bit = 1 sec */
/*****************************************************************************/
#define      _BLFR 0x08    /* Backlight fade-in and fade-out times */
struct reg_0x08     {
      unsigned BL_FI : 4;
      unsigned BL_FO : 4;
                    };
/*****************************************************************************/
#define _BLMX1 0x09        /* Backlight MAX brightness for daylight condition */
unsigned char blmx1 = 0;   /* Value range 0 to 127, 1-bit value depends on law used (linear or
square) */
/*****************************************************************************/
#define _BLMD1 0x0A        /* Backlight DIM brightness for daylight condition */
unsigned char blmd1 = 0;   /* Value range 0 to 127, 1-bit value depends on law used (linear or
square) */
/*****************************************************************************/
#define _BLMX2 0x0B        /* Backlight MAX brightness for office condition */
unsigned char blmx2 = 0;   /* Value range 0 to 127, 1-bit value depends on law used (linear or
square) */
/*****************************************************************************/
#define _BLMD2 0x0C        /* Backlight DIM brightness for office condition */
unsigned char blmd2 = 0;   /* Value range 0 to 127, 1-bit value depends on law used (linear or
square) */
```

```
/*****************************************************************************/
#define _BLMX3 0x0D        /* Backlight MAX brightness for dark condition */
unsigned char blmx3 = 0;   /* Value range 0 to 127, 1-bit value depends on law used (linear or
square) */
/*****************************************************************************/
#define _BLMD3 0x0E        /* Backlight DIM brightness for dark condition */
unsigned char blmd3 = 0;   /* Value range 0 to 127, 1-bit value depends on law used (linear or
square) */
/*****************************************************************************/
#define _ISCFR 0x0F        /* Independent sinks current fade law */
unsigned char sc_law = 0;  /* Value 0 to 3 (0 = linear law, 1 = square law, 2 = square law Cubic
1, 3 = square law Cubic 2) */
/*****************************************************************************/
#define _ISCC 0x10         /* Independent sinks control register */
struct reg_0x10 {
      unsigned SC1_EN : 1;
      unsigned SC2_EN : 1;
      unsigned SC3_EN : 1;
      unsigned SC4_EN : 1;
      unsigned SC5_EN : 1;
      unsigned SC6_EN : 1;
      unsigned SC7_EN : 1;
             };
/*****************************************************************************/
#define      _ISCT1 0x11   /* Independent sinks on time and LED7 to LED5 off-time control */
struct reg_0x11     {
      unsigned SC5OFF :2;
      unsigned SC6OFF : 2;
      unsigned SC7OFF : 2;
      unsigned SCON : 2;
                          };
/*****************************************************************************/
#define _ISCT2 0x12        /* Independent sinks LED4 to LED1 off-time control */
struct reg_0x12     {
      unsigned SC1OFF : 2;
      unsigned SC2OFF : 2;
      unsigned SC3OFF : 2;
      unsigned SC4OFF : 2;
                          };
/*****************************************************************************/
#define _ISCF 0x13         /* Independent sinks fade control */
struct reg_0x13     {
      unsigned SCFI : 4;
      unsigned SCFO : 4;
                    };
```

```
/******************************************************************************/
#define _ISC7 0x14         /* Independent sink current for LED7 */
struct reg_0x14     {
       unsigned SCD7 : 7;  /* Value range 0 to 127, 1-bit value depends on use */
       unsigned SCR : 1;
                    };
/******************************************************************************/
#define _ISC6 0x15         /* Independent sink current for LED6 */
unsigned char isc6 = 0;    /* Value range 0 to 127, 1-bit value depends on use */
/******************************************************************************/
#define _ISC5 0x16         /* Independent sink current for LED5 */
unsigned char isc5 = 0;    /* Value range 0 to 127, 1-bit value depends on use */
/******************************************************************************/
#define _ISC4 0x17         /* Independent sink current for LED4 */
unsigned char isc4 = 0;    /* Value range 0 to 127, 1-bit value depends on use */
/******************************************************************************/
#define _ISC3 0x18         /* Independent sink current for LED3 */
unsigned char isc3 = 0;    /* Value range 0 to 127, 1-bit value depends on use */
/******************************************************************************/
#define _ISC2 0x19         /* Independent sink current for LED2 */
unsigned char isc2 = 0;    /* Value range 0 to 127, 1-bit value depends on use */
/******************************************************************************/
#define _ISC1 0x1A         /* Independent sink current for LED1 */
unsigned char isc1 = 0;    /* Value range 0 to 127, 1-bit value depends on use */
/******************************************************************************/
#define _CCFG 0x1B         /* Light sensing Comparator 1 configuration register */
struct reg_0x1B {
       unsigned L2_EN : 1;
       unsigned L3_EN : 1;
       unsigned L2_OUT : 1;
       unsigned L3_OUT :1;
       unsigned FORCE_RD : 1;
       unsigned FILT : 3;
                    };
/******************************************************************************/
#define _CCFG2 0x1C        /* Light sensing Comparator 2 configuration register */
struct reg_0x1C {
       unsigned L2_EN2 : 1;
       unsigned L3_EN2 : 1;
       unsigned L2_OUT2 : 1;
       unsigned L3_OUT2 :1;
       unsigned FORCE_RD2 : 1;
       unsigned FILT2 : 1;
                    };
```

```
/*****************************************************************************/
#define _L2TRP 0x1D       /* L2 brightness comparator reference for both phototransistors */
unsigned char l2trp = 0;
/*****************************************************************************/
#define _L2HYS 0x1E       /* L2 comparator hysteresis for both phototransistors */
unsigned char l2hys = 0;
/*****************************************************************************/
#define _L3TRP 0x1F       /* L3 brightness comparator reference for both phototransistors */
unsigned char l3trp = 0;
/*****************************************************************************/
#define _L3HYS 0x20       /* L3 comparator hysteresis for both phototransistors */
unsigned char l3hys = 0;
/*****************************************************************************/
#define _PH1LEVL 0x21     /* First phototransistor ambient light level-lower byte */
unsigned char ph1levl = 0;
/*****************************************************************************/
#define _PH1LEVH 0x22     /* First phototransistor ambient light level-upper byte */
unsigned char ph1levh = 0;
/*****************************************************************************/
#define _PH2LEVL 0x23     /* Second phototransistor ambient light level-lower byte */
unsigned char ph2levl = 0;
/*****************************************************************************/
#define _PH2LEVH 0x24     /* Second phototransistor ambient light level-upper byte */
unsigned char ph2levh = 0;
/*****************************************************************************/
```

LISTING FOR ADP8860 CONSTANTS.H

```
const unsigned char ENABLE_BACKLIGHT = 1;
const unsigned char DISABLE_BACKLIGHT = 0;
const unsigned char LIGHT_SENSOR_AUTO = 1;
const unsigned char LIGHT_SENSOR_MANUAL = 0;
const unsigned char ENABLE_SINKS_SYNCHRONOUSLY = 1;
const unsigned char DISABLE_SINKS_SYNCHRONOUSLY = 0;
const unsigned char FORCE_DIM_LEVEL = 1;
const unsigned char CONTRL_DIM_LEVEL = 0;
const unsigned char NORMAL_MODE = 1;
const unsigned char STANDBY_MODE = 0;
/*********************************/
const unsigned char ENABLE_INTERRUPT = 1;
const unsigned char DISABLE_INTERRUPT = 0;
/*********************************/
const unsigned char OVERRIDE_BACKLIGHT_FADEIN = 1;
const unsigned char ENABLE_BACKLIGHT_FADEIN = 0;
const unsigned char BL_LINEAR_LAW = 0;
const unsigned char BL_SQUARE_LAW = 1;
const unsigned char BL_CUBIC1_LAW = 2;
const unsigned char BL_CUBIC2_LAW = 3;
const unsigned char FORCE_BACKLIGHT_DAYLIGHT = 0;
const unsigned char FORCE_BACKLIGHT_OFFICE = 1;
const unsigned char FORCE_BACKLIGHT_DARK = 2;
const unsigned char DISABLE_FORCE_BACKLIGHT = 3;
const unsigned char D6_IS_2nd_PHOTOSENSOR = 1;
const unsigned char D6_IS_CURRENT_SINK = 0;
const unsigned char SELECT_2nd_PHOTOSENSOR = 1;
const unsigned char SELECT_1st_PHOTOSENSOR = 0;
/*********************************/
const unsigned char LED_INDEPENDENT_SINK = 1;
const unsigned char LED_IS_BACKLIGHT = 0;
/********************************/
const unsigned char FADE_DISABLE = 0;
const unsigned char FADING_0p3SEC = 1;
const unsigned char FADING_0p6SEC = 2;
const unsigned char FADING_0p9SEC = 3;
const unsigned char FADING_1p2SEC = 4;
const unsigned char FADING_1p5SEC = 5;
const unsigned char FADING_1p8SEC = 6;
const unsigned char FADING_2p1SEC = 7;
const unsigned char FADING_2p4SEC = 8;
const unsigned char FADING_2p7SEC = 9;
const unsigned char FADING_3p0SEC = 10;
const unsigned char FADING_3p5SEC = 11;
const unsigned char FADING_4p0SEC = 12;
const unsigned char FADING_4p5SEC = 13;
```

```
const unsigned char FADING_5p0SEC = 14;
const unsigned char FADING_5p5SEC = 15;
/********************************/
const unsigned char SINK_LINEAR_LAW = 0;
const unsigned char SINK_SQUARE_LAW = 1;
const unsigned char SINK_CUBIC1_LAW = 2;
const unsigned char SINK_CUBIC2_LAW = 3;
/********************************/
const unsigned char ENABLE_ISINK = 1;
const unsigned char DISABLE_ISINK = 0;
/********************************/
const unsigned char ISINK_0p2SEC_ON = 0;
const unsigned char ISINK_0p6SEC_ON = 1;
const unsigned char ISINK_0p9SEC_ON = 2;
const unsigned char ISINK_1p2SEC_ON = 3;
const unsigned char ISINK_ALWAYS_ON = 0;
const unsigned char ISINK_0p6SEC_OFF = 1;
const unsigned char ISINK_1p2SEC_OFF = 2;
const unsigned char ISINK_1p8SEC_OFF = 3;
/********************************/
const unsigned char SINK7_60mA = 1;
const unsigned char SINK7_30mA = 0;
const float FULL_SCALE_30mA = 30;
const float FULL_SCALE_60mA = 60;
/********************************/
const unsigned char L2_AUTO_LEVEL = 1;
const unsigned char L2_COMP_DISABLED = 0;
const unsigned char L3_AUTO_LEVEL = 1;
const unsigned char L3_COMP_DISABLED = 0;
const unsigned char FORCE_COMP_READ = 1;
const unsigned char LIGHTSENS_FILTER_80mS = 0;
const unsigned char LIGHTSENS_FILTER_160mS = 1;
const unsigned char LIGHTSENS_FILTER_320mS = 2;
const unsigned char LIGHTSENS_FILTER_640mS = 3;
const unsigned char LIGHTSENS_FILTER_1280mS = 4;
const unsigned char LIGHTSENS_FILTER_2560mS = 5;
const unsigned char LIGHTSENS_FILTER_5120mS = 6;
const unsigned char LIGHTSENS_FILTER_10240mS = 7;
/********************************/
const float L2_LUX_X_LSB = 10;
const float L2_uA_X_LSB = 4;
const float L3_LUX_X_LSB = 1.25;
const float L3_uA_X_LSB = 0.5;
/********************************/
```

LISTING FOR ADP8860 RESET.H

```
/********************************************************************************/
struct reg_0x00 mfdvid;
/********************************************************************************/
struct reg_0x01 mdcr = {0,0,0,0,0,0};
/********************************************************************************/
struct reg_0x02 mdcr2 = {0,0,0,0,0};
/********************************************************************************/
struct reg_0x03 intr_en = {0,0,0,0,0};
/********************************************************************************/
struct reg_0x04 cfgr = {0,0,0,0,0};
/********************************************************************************/
struct reg_0x05 blsen = {0,0,0,0,0,0,0};
/********************************************************************************/
struct reg_0x08 blfr = {0,0};
/********************************************************************************/
struct reg_0x10 iscc = {0,0,0,0,0,0,0};
/********************************************************************************/
struct reg_0x11 isct1 = {0,0,0,0};
/********************************************************************************/
struct reg_0x12 istc2 = {0,0,0,0};
/********************************************************************************/
struct reg_0x13 iscf = {0,0};
/********************************************************************************/
struct reg_0x14 isc7 = {0,0};
/********************************************************************************/
struct reg_0x1B ccfg = {0,0,0,0,0,0};
/********************************************************************************/
struct reg_0x1C ccfg2 = {0,0,0,0,0,0};
/***********************************
```

LISTING FOR ADP8860 MAINPROGRAM.C

```
#include <stdio.h>
#include <tcconio.h>
#include <string.h>
#include <math.h>
#include "ADP8860 regs.h"
#include "ADP8860 constants.h"
#include "ADP8860 reset.h"
const unsigned char SLAVE_ID_WR = 0x54;
const unsigned char SLAVE_ID_RD = 0x55;
/*********************************************************************
* This main program simulates the ADP8860 registers initialization
* using the following conditions:
*      LED1 to LED4 are part of the backlight
*      LED5 and LED7 are individual sinks.
*      Backlight current is controlled automatically by two comparators
*      The brightness levels in the three modes are as follows:
*             Dark: MAX = 13.5 mA - DIM = 3.5 mA
*             Office: MAX = 20.7 mA - DIM = 5 mA
*             Daylight: MAX = 27.4 mA - DIM = 8.5 mA
*      Backlight fade-in time is 1.5 sec
*      Backlight fade-out time is 4 sec
*      Time backlight driven at MAX level is 30 sec
*      Time backlight driven at DIM level is 15 sec
*      Current for LED5 is 20 mA
*      Current for LED7 is 55 mA (drives keypad light)
*      Individual sinks fade-in time is 0.6 sec
*      Individual sinks fade-out time is 4 sec
*      LED5, when activated, is turned on for 0.6 sec and off for 1.8 sec
*      LED7, when activated, is always on
*      Dark-to-office comparator threshold (L3) is 100 LUX
*      L3 comparator hysteresis is +15 LUX (115 LUX)
*      Office to daylight comparator threshold (L2) is 700 LUX
*      L2 comparator hysteresis is +100 LUX (800 LUX)
*      The values given before do not represent the real operating conditions
*      and are only provided as an indicative example to set up the device operation
*********************************************************************/
```

```
int main(void)
{
       clrscr();
/* Control Registers and Interrupt Initialization */
       mdcr.CMP_AUTOEN = LIGHT_SENSOR_AUTO;  /* Enable backlight auto control from comparator */
       cfgr.LAW = BL_SQUARE_LAW;             /* Default transfer law for setting backlight LED current */
       cfgr.CMP2_SEL = D6_IS_2nd_PHOTOSENSOR; /* Select D6 (LED6) as input for 2nd phototransistor */
       cfgr.BLV = DISABLE_FORCE_BACKLIGHT;   /* Backlight levels are not software forced */
       cfgr.FOVR = ENABLE_BACKLIGHT_FADEIN;  /* Allows backlight fade-in time */
       intr_en.CMP_IEN = DISABLE_INTERRUPT;  /* Disable interrupts from Comparator 1 */
       intr_en.CMP2_IEN = DISABLE_INTERRUPT; /* Disable interrupts from Comparator 2 */
       intr_en.OVP_IEN = ENABLE_INTERRUPT;   /* Enable interrupt caused by overvoltage conditions */
       intr_en.TSD_IEN = ENABLE_INTERRUPT;   /* Enable interrupt caused by thermal shutdown */
       intr_en.SHORT_IEN = ENABLE_INTERRUPT; /* Enable interrupt caused by short circuits */
/* Backlight Parameters Initialization */
       blsen.D5EN = LED_INDEPENDENT_SINK;    /* LED 5 selected as individual sink */
       blsen.D7EN = LED_INDEPENDENT_SINK;    /* LED 7 selected as individual sink */
       dimt = 30;                            /* Backlight timeout in MAX phase is 30 sec */
       offt = 15;                            /* Backlight timeout in DIM phase is 15 sec */
       blfr.BL_FI = FADING_1p5SEC;           /* Backlight fade-in time is 1.5 sec */
       blfr.BL_FO = FADING_4p0SEC;           /* Backlight fade-out time is 4 sec */
       const float BLMAX_DAYLIGHT_CURRENT_mA = 27.4; /* Default MAX daylight backlight desired current in mA */
       const float BLMAX_OFFICE_CURRENT_mA = 20.7;    /* Default MAX office backlight desired current in mA */
       const float BLMAX_DARK_CURRENT_mA = 13.5;      /* Default MAX dark backlight desired current in mA */
       const float BLDIM_DAYLIGHT_CURRENT_mA = 8.5;   /* Default DIM daylight backlight desired current in mA */
       const float BLDIM_OFFICE_CURRENT_mA = 5;       /* Default DIM office backlight desired current in mA */
       const float BLDIM_DARK_CURRENT_mA = 3.5;       /* Default DIM dark backlight desired current in mA */
       blmx1 = CalcCurrent(BLMAX_DAYLIGHT_CURRENT_mA,cfgr.LAW,FULL_SCALE_30mA); /* Calculate the brightness
register value corresponding to the desired backlight current */
       blmx2 = CalcCurrent(BLMAX_OFFICE_CURRENT_mA,cfgr.LAW,FULL_SCALE_30mA);
       blmx3 = CalcCurrent(BLMAX_DARK_CURRENT_mA,cfgr.LAW,FULL_SCALE_30mA);
       blmd1 = CalcCurrent(BLDIM_DAYLIGHT_CURRENT_mA,cfgr.LAW,FULL_SCALE_30mA);
       blmd2 = CalcCurrent(BLDIM_OFFICE_CURRENT_mA,cfgr.LAW,FULL_SCALE_30mA);
       blmd3 = CalcCurrent(BLDIM_DARK_CURRENT_mA,cfgr.LAW,FULL_SCALE_30mA);
/* Individual Sinks Parameters Initialization */
    sc_law = SINK_SQUARE_LAW;                     /* Individual sink transfer law is square */
       isct1.SCON = ISINK_0p6SEC_ON;              /* Enabled individual sinks on for 0.6 sec */
       isct1.SC7OFF = ISINK_ALWAYS_ON;            /* LED7 when activated is always on (keypad light) */
       isct1.SC5OFF = ISINK_1p8SEC_OFF;           /* LED5 when activated stays on 0.6 sec and off 1.8 sec */
       iscf.SCFI = FADING_0p6SEC;                 /* Fade-in time for the individual sinks is 0.6 sec */
       iscf.SCFO = FADING_4p0SEC;                 /* Fade-out time for the individual sinks is 4 sec */
       const float SC5MAX_CURRENT_mA = 20;        /* Default max current for LED5 in mA */
       const float SC7MAX_CURRENT_mA = 55;        /* Default max current for LED7 in mA */
       isc5 = CalcCurrent(SC5MAX_CURRENT_mA,sc_law,FULL_SCALE_30mA);
       isc7.SCD7 = CalcCurrent(SC7MAX_CURRENT_mA,sc_law,FULL_SCALE_60mA);
       isc7.SCR = SINK7_60mA;
```

```
/* Comparator Parameters Initialization */
        ccfg.FILT = LIGHTSENS_FILTER_1280mS;           /* Select 1.28 sec filter */
        ccfg.FILT2 = LIGHTSENS_FILTER_1280mS;          /* Select 1.28 sec filter */
        ccfg.L2_EN = L2_AUTO_LEVEL;                    /* L2 Comparator 1 allowed to auto control backlight */
        ccfg.L3_EN = L3_AUTO_LEVEL;                    /* L3 comparator 1 allowed to auto control backlight */
        ccfg2.L2_EN2 = L2_AUTO_LEVEL;                  /* L2 comparator 2 allowed to auto control backlight */
        ccfg2.L3_EN2 = L3_AUTO_LEVEL;                  /* L3 comparator 2 allowed to auto control backlight */
        l2trp = rint(700/L2_LUX_X_LSB);                /* Set L2 trip point threshold */
        l2hys = rint(100/L2_LUX_X_LSB);                /* Set L2 hysteresis threshold, this value is added to
l2trp level */
        l3trp = rint(100/L3_LUX_X_LSB);                /* Set L2 trip point threshold */
        l3hys = rint(15/L3_LUX_X_LSB);                 /* Set L3 hysteresis point threshold, this value is
added to l3trp level */
        seam_read = ENABLE_SEAMLESS;                   /* Enable special Motorola seamless mode */
/* Simulate Commands Sent Out Through the I2C Interface */
        printf("- (I2C Write) DevID+W = 0x%X [RegAddr = 0x%X], [Value = 0x%X]\n", SLAVE_ID_WR,_MDCR,mdcr);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_CFGR,cfgr);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value =
0x%X]\n",SLAVE_ID_WR,_INTR_EN,intr_en);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_BLSEN,blsen);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_BLOFF,offt);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_BLDIM,dimt);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_BLFR,blfr,0);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_BLMX1,blmx1);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_BLMD1,blmd1);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_BLMX2,blmx2);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_BLMD2,blmd2);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_BLMX3,blmx3);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_BLMD3,blmd3);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_ISCFR,sc_law);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_ISCT1,isct1);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_ISCF,iscf);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_ISC5,isc5);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_ISC7,isc7);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_CCFG,ccfg);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_CCFG2,ccfg2);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_L2TRP,l2trp);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_L2HYS,l2hys);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_L3TRP,l3trp);
        printf("- (I2C Write) DevID+W = 0x%X,[RegAddr = 0x%X], [Value = 0x%X]\n",SLAVE_ID_WR,_L3HYS,l3hys);
        return 0;
}
```

```
/*********************************************************************************************
Support Functions
*********************************************************************************************/
/* Returns the LED current value based on the value written in the register, the transfer law, and the full-
scale current value */
float GetCurrent(int CurrVal, int Law, int FullScale )
{
  if (Law == 1)
          return (CurrVal*0.236);
  else return pow((CurrVal*sqrt(FullScale)/127),2);
}
/*******************************************************************************************************/
/* Returns the value to write in the LED current register based on the desired current value, transfer law,
and full-scale current value */
unsigned char CalcCurrent(float DesiredCurrent, int Law, float FullScale)
{
  if (Law == 0)
          return rint((DesiredCurrent * 127)/FullScale);
  else return rint(127*sqrt(DesiredCurrent)/sqrt(FullScale));
}
```

NOTES

NOTES